An Efficient Hypergraph Approach to Robust Point Cloud Resampling

Qinwen Deng, *Student Member, IEEE*, Songyang Zhang, and Zhi Ding, *Fellow, IEEE*

***Abstract*—Efficient processing and feature extraction of large-scale point clouds are important in related computer vision and cyber-physical systems. This work investigates point cloud resampling based on hypergraph signal processing (HGSP) to better explore the underlying relationship among different points in the point cloud and to extract contour-enhanced features. Specifically, we design hypergraph spectral filters to capture multilateral interactions among the signal nodes of point clouds and to better preserve their surface outlines. Without the need and the computation to first construct the underlying hypergraph, our low complexity approach directly estimates hypergraph spectrum of point clouds by leveraging hypergraph stationary processes from the observed 3D coordinates. Evaluating the proposed resampling methods with several metrics, our test results validate the high efficacy of hypergraph characterization of point clouds and demonstrate the robustness of hypergraph-based resampling under noisy observations.**

***Index Terms*—Hypergraph signal processing, compression, point cloud resampling, virtual reality.**

I. Introduction

3D perception plays an important role in the high growth fields of robotics and cyber-physical systems and continues to drive many progresses made in advanced point cloud processing. 3D point clouds provide efficient exterior representation for complex objects and their surroundings. Point clouds have seen broad applications in many areas, such as computer vision, autonomous driving and robotics. Notable examples of point cloud processing include surface reconstruction [1], rendering [2], feature extraction [3], shape classification [4], and object detection/tracking [5]. When constructing point cloud of a target object, however, modern laser scan systems can generate millions of data points [6]. To achieve better storage efficiency and lower point cloud processing complexity, point cloud resampling aims to reduce the number of points in a cloud to achieve data compression while preserving the vital 3D structural and surface features. Point cloud resampling represents an important tool in various applications such as point cloud segmentation, object classification and efficient data representation. An example of point cloud resampling proposed in [7] suggested a graph-based filter to downsample point clouds and to capture the original object surface contour.

The literature already contains a variety of works on different aspects of point cloud resampling. For instance, a centroidal Voronoi tessellation method in [8] can progressively generate high-quality resampling results with isotropic or anisotropic distributions from a given point cloud to form compact representations of the underlying cloud surface. Another 3D filtering and downsampling technique [9] relies on a growing neural gas network, to deal with noise and outliers within data provided by Kinect sensors. Of particular interest is graph-based resampling approach which has exhibited desirable capability to capture the underlying structures of point clouds [10]. The graph-based method of [12] applies embedded binary trees to compress the dynamic point cloud data. Another interesting work [7] proposes several graph-based filters to capture the distribution of point data to achieve computationally efficient resampling. In addition, a contour-enhanced resampling method introduced in [11] utilizes graph-based highpass filters.

In addition to the aforementioned class of graph-based methods, a competing class of feature-based approaches via edge detection and feature extraction has also been popular. In [13], the authors presented a sharp feature detector via Gaussian map clustering on local neighborhoods. Bazazian *et al.* [14] extended this principle by leveraging principal component analysis (PCA) to develop a new agglomerative clustering method. The efficiency and accuracy of this work can further benefit from spectral analysis of the covariance matrix defined by $k-$nearest neighbors. Another typical approach represented by [15] processes a noisy and possibly outlier-ridden point set in an edge-aware manner.

Both graph-based and feature-based methods have clearly achieved successes in point cloud resampling. However, some limitations remain. Graph-based methods tend to focus on pairwise relationship between different points, since each graph edge only connects two signal nodes. However, it is clear that multilateral interactions of data points could model the more informative characteristics of 3D point clouds. Bilateral graph node connections cannot even describe multilateral relationship among points on the same surface (e.g. 3 points of a triangle) directly [23]. Furthermore, in graph-based methods, efficient construction of a suitable graph to represent an arbitrary point cloud poses another challenge. Among feature-based methods, performance would vary with respect to the feature selection and filter designs. The open issues are the adequate and robust selection of features and filter parameters for practical point cloud processing.

More recently, hypergraphs have been successfully applied in representing and characterizing the underlying multilateral interactions among multimedia data points [16]. A hypergraph extends basic graph concept into higher dimensions,

This material is based upon work supported by the National Science Foundation under Grant No. 1824553, Grant No. 2029027, and Grant No. 2029848.

Q. Deng, S. Zhang, and Z. Ding are with Department of Electrical and Computer Engineering, University of California, Davis, CA, 95616. (E-mail: mrdeng@ucdavis.edu, sydzhang@ucdavis.edu, and zding@ucdavis.edu).

(a) Original. (b) Resampled.

Fig. 1. Example of Contour-Enhanced Resampling: (a) original point cloud with 272705 points, and (b) resampled building based on the proposed local hypergraph filtering with 20% samples.

in which each hyperedge can connect more than two nodes [15]. Therefore for point clouds, hypergraph provides a more general representation to characterize multilateral relationship for points on object surfaces such that one hyperedge can cover multiple nodes on the same surface. Furthermore, by generalizing graph signal processing [19], hypergraph signal processing (HGSP) [21] [20] provides a theoretical foundation for spectral analysis in hypergraph-based point cloud processing. Specifically, stationarity-based hypergraph estimation, in conjunction with hypergraph-based filters, has demonstrated notable successes in processing point clouds for various tasks including segmentation, sampling, and denoising [22], [23].

In this paper, we investigate point cloud resampling based on hypergraph spectral analysis. Instead of the traditional uniform resampling, we investigate contour-enhanced resampling to select subset of points in the point cloud and to extract distinct surface features. A heuristic example is illustrated in Fig. 1 showing a point cloud successfully resampled with only 20% samples for the building. To briefly highlight the novelty of our proposed approaches, we estimate the hypergraph spectrum basis for point clouds under study by leveraging the hypergraph stationary process. We propose three novel 3D point cloud resampling methods:

1) Hypergraph kernel convolution method (HKC);
2) Hypergraph kernel filtering method (HKF);
3) Local hypergraph filtering method (LHF).

The kernel convolution method defines a local smoothness among signals based on an operator and hypergraph convolution. The kernel filtering method defines the local smoothness with respect to highpass filtering in spectrum domain. The local hypergraph filtering method utilizes a local sharpness definition with respect to highpass filtering in spectrum domain. In order to test the model preserving property on complex point cloud models, we apply a simple method for point cloud recovery based on alpha complex and Poisson sampling. We then test the proposed methods under several metrics to demonstrate the compression efficiency and robustness of our proposed resampling methods with respect to the general feature preservation of point clouds under study.

We summarize the major contributions of this manuscript:

- We propose three novel hypergraph-based resampling methods to preserve distinct and sharp point cloud features;
- We provide novel definitions of hypergraph-based indicators to evaluate the smoothness or sharpness over point clouds;
- We apply different metrics to demonstrate the effectiveness of our proposed methods.

We organize the rest of the manuscript as follows. Section II briefly describe basic point cloud model and introduces the fundamentals of hypergraph signal processing. We develop the foundation of HGSP based point cloud resampling and derive three new resampling methods in Section III. We provide the test results of the proposed resampling methods in Section IV, before formalizing our conclusions in Section V.

II. Fundamentals and Background

A. Point Cloud

A point cloud is a collection of points on the surface of a 3D target object. Each point consists of its 3D coordinates and may contain further features, such as colors and normals [28]. In this work, we focus on the coordinates of data points and point cloud resampling. A point cloud can be represented by the coordinates of N data points written as an $N \times 3$ real-valued location matrix

$$\mathbf{P} = [\mathbf{X_1} \quad \mathbf{X_2} \quad \mathbf{X_3}] = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \cdots \\ \mathbf{p}_N^T \end{bmatrix} \in \mathbb{R}^{N \times 3}, \tag{1}$$

where $\mathbf{X}_i$ denotes a vector of the i-th coordinates of all N data points whereas 3×1 vector $\mathbf{p}_i$ indicates the i-th point's coordinates.

B. Hypergraph Signal Processing

Hypergraph signal processing (HGSP) is an analytic framework that uses hypergraph and tensor representation to model high-order signal interactions [21]. Within this framework, a Mth-order N-dimensional representation tensor $\mathbf{A} = (a_{i_1 i_2 \cdots i_M}) \in \mathbb{R}^{N^M}$ models a hypergraph with N vertices in each hyperedge, which is capable of connecting maximum of M nodes. We may call the number of nodes connected by a hyperedge as its length. Weights of hyperedges with length less than M are normalized according to combinations and permutations [18].

Orthogonal CANDECOMP/PARAFAC (CP) decomposition [24]- [26] enables the (approximate) decomposition of a representing tensor into

$$\mathbf{A} \approx \sum_{r=1}^{N} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{M \text{ times}}, \tag{2}$$

where $\circ$ denotes tensor outer product, $\{\mathbf{f}_1, \cdots, \mathbf{f}_N\}$ are orthonormal basis to represent spectrum components, and λ_r is the r-th spectrum coefficient corresponding to the r-th basis. Spectrum components $\{\mathbf{f}_1, \cdots, \mathbf{f}_N\}$ form the full hypergraph spectral space.

Similar to GSP, hypergraph signals are attributes of nodes. Intuitively, a signal is defined as $\mathbf{s} = [s_1 \quad s_2 \quad ... \quad s_N]^{\mathrm{T}} \in \mathbb{R}^N$. Since the adjacency tensor $\mathbf{A}$ describes high-dimensional interactions of signals, we define a special form of the hypergraph signal to work with the representing tensor, i.e.,

$$\mathbf{s}^{[M-1]} = \underbrace{\mathbf{s} \circ ... \circ \mathbf{s}}_{\text{M-1 times}}. \tag{3}$$

Given the definitions of hypergraph spectrum and hypergraph signals, hypergraph Fourier transform (HGFT) is given by

$$\hat{\mathbf{s}} = \mathcal{F}_C(\mathbf{s}) = [(\mathbf{f}_1^{\mathrm{T}}\mathbf{s})^{M-1} \cdots (\mathbf{f}_N^{\mathrm{T}}\mathbf{s})^{M-1}]^{\mathrm{T}}. \tag{4}$$

From the graph specific HGFT, hypergraph spectral convolution can be generalized [16] as

$$\mathbf{x} \diamond \mathbf{y} = \mathcal{F}_C^{-1}(\mathcal{F}_C(\mathbf{x}) \odot \mathcal{F}_C(\mathbf{y})), \tag{5}$$

where $\mathcal{F}_C$ is the HGFT, $\mathcal{F}_C^{-1}$ denotes inverse HGFT (iHGFT), and $\odot$ denotes Hadamard product [21]. This definition applies the basic relationship between convolution and spectrum product, and generalizes convolution in the vertex domain into product in the hypergraph spectrum domain.

To be concise, we refrain from a full review of HGSP here. Instead, we refer readers to [21] and related works for a more extensive introduction of HGSP concepts, such as filtering, hypergraph Fourier transform, and sampling theory, among others. Equally important are concept and properties of hypergraph stationary processes which can be found in, e.g., [23].

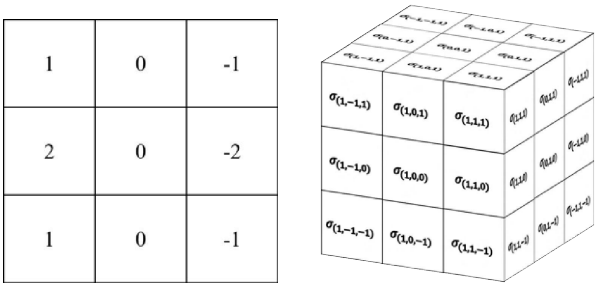
III. HGSP Point Cloud Resampling

Within the HGSP framework, we now propose three novel edge-preserving methods for point cloud resampling. First, we develop a hypergraph kernel convolution (HKC) method inspired by the convolution-based edge detection in image processing. Next, we propose a hypergraph kernel filtering (HKF) algorithm targeting local smoothness to reduce the computational complexity of HKC. Finally, we design a local hypergraph filter (LHF) to target point clouds with non-uniformly distributed points over the surface of an object.

A. Hypergraph Kernel Convolution (HKC) based Method

In traditional image processing, kernel convolution methods such as Sobel and Prewitt [27] have achieved notable successes in edge detection. Inspired by these 2D kernel convolution methods in 2D image processing, e.g. Fig. 2(a), we define a square $k \times k \times k$ 3D cube as the slicing block to define a local signal $\mathbf{s}_i \in \mathbb{R}^{N_k}$ and a convolution kernel $\mathbf{G} \in \mathbb{R}^{N_k}$ with $N_k = k^3$ aimed at extracting sharp outliers of the point cloud under study. An example of 3^3 cubic 3D convolution kernel is shown in Fig. 2(b). Note that the hypergraph convolution kernel can assume different shapes and sizes depending on the datasets.

For the i-th point in an original point cloud, its corresponding local signal $\mathbf{s}_i \in \mathbb{R}^{N_k}$ is defined according to the number of points in the voxel of kernel centered at i-th point. An example of the local signal is shown in Fig. 3. Although the idea behind the use of, e.g., 3D Sobel operator



(a) 2D Sobel Kernel. (b) 3D Hypergraph Kernel.

Fig. 2. Example of Convolution Kernels.

is straightforward, technical obstacles arise mainly due to two reasons: (1) nodes in 3D point cloud are not always on grid; (2) two signals in graph/hypergraph based convolution must have the same length. Thus, we let N_k be equal to the number of voxels in the kernel. Let d be the distance between the centers of two nearby voxels in the kernel. A proper selection of d should allow $\mathbf{s}_i$ to capture the local geometric information. If d is too small, only a few neighbors of i-th point are included in the $\mathbf{s}_i$ and it is sensitive to measurement noise; If d is too large, large number of neighbors are included in each voxel, which may lead to the blurring of detailed local geometric information. Since intrinsic resolution describes the point cloud density and can be estimated by averaging all distances between each point and its nearest neighbor, we set d as the intrinsic resolution of a point cloud.

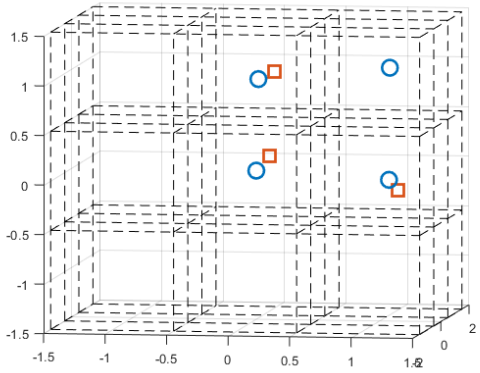


Fig. 3. An example of local signal $\mathbf{s}_i$: Points in the middle layer are marked by blue circles and points in the back layer are marked by red squares. The voxels of slicing block are delineated by the dash black lines. There is at most one point in each voxel in this example, such that $\mathbf{s}_i(n) = 1$ or 0, respectively, depending on whether the n-th voxel contains a point.

Given the definition of signals, we now propose a new hypergraph convolution based method. There are two main steps in our kernel convolution based method: 1) estimation of hypergraph spectrum; and 2) implementation of kernel convolution.

We first estimate the hypergraph spectrum based on hypergraph stationary process. Introduced in [23], a stochastic signal $\mathbf{x}$ is weak-sense hypergraph stationary if and only if it has zero-mean and its covariance matrix has the same eigenvectors

as the hypergraph spectrum basis, i.e.,

$$\mathbb{E}[\mathbf{x}] = \mathbf{0}, \tag{6}$$

and

$$\mathbb{E}[\mathbf{x}\mathbf{x}^H] = \mathbf{V}\Sigma_{\mathbf{x}}\mathbf{V}^H, \tag{7}$$

where $\mathbf{V}$ is the hypergraph spectrum basis. Since the 3D coordinates can be viewed as three observations of the point cloud from different projection directions, we can estimate the hypergraph spectrum from the covariance matrix of three coordinates based on assumption of signal stationarity. Note that, since GSP is a special of HGSP when the number of edges is reduced to two, HGSP-based stationary process is also compatible with GSP-based stationarity. Here, we use the same spectrum estimation strategy as [23], and consider the adjacency tensor as a third-order tensor, which is the minimum number of nodes to form a surface. Interested reader can refer to [22], [23] for more detailed discussions on the spectrum estimation based on HGSP-based stationary process

1) HKC Algorithm: Let $\mathbf{P_c} = [\mathbf{X_1}\ \mathbf{X_2}\ \mathbf{X_3}] \in \mathbb{R}^{N_k \times 3}$ be the coordinates of the total N_k voxel centers in the kernel. For our $3\times3\times3$ kernel example, $N_k = 27$. The distance d is set to equal the intrinsic resolution of the point cloud, which can be estimated by the average of distances between each point and its nearest neighbor in the point cloud. We then normalize the coordinates $\mathbf{P_c}$ to obtain a zero mean signal $\mathbf{P'_c}$. By calculating the eigenvectors $\{\mathbf{f}_1, \cdots, \mathbf{f}_{N_k}\}$ for $R_{\mathbf{P'_c}} = \mathbf{P'_c}\mathbf{P'_c}^T$, we can estimate the hypergraph spectrum basis $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_{N_k}]$.

Next, we implement convolution between the local signal $\mathbf{s}_i$ and the kernel $\mathbf{G}$. Since we consider the third-order tensor and are only interested in signal energies in the spectrum domain, we can utilize the hypergraph spectrum to calculate a simplified-form of HGFT $\hat{\mathbf{s}}_i$ and a corresponding inverse HGFT (iHGFT) of original signals $\mathbf{s}_i$, instead of the HGFT and iHGFT of the hypergraph signal $\mathbf{s}_i^{[M-1]}$ in (4).

Recall from [21] that simplified HGFT and iHGFT of original signal are, respectively,

$$\mathcal{F}(\mathbf{s}_i) = \hat{\mathbf{s}}_i = \mathbf{V}^H\mathbf{s}_i, \tag{8}$$

$$\mathcal{F}^{-1}(\hat{\mathbf{s}}_i) = \mathbf{s}_i = \mathbf{V}\hat{\mathbf{s}}_i. \tag{9}$$

Note that the hypergraph signal is a $(M-1)$-fold tensor outer product of the original signal in vertex domain, corresponding to $(M-1)$-fold Hadamard product in spectrum domain, where they share the same bandwidth.

Directly designing convolution kernel $\mathbf{G}$ in vertex domain is challenging for two main reasons. First, since hypergraph spectrum basis $\mathbf{V}$ would vary for different kernels, finding a general $\mathbf{G}$ in vertex domain that performs equally well for various different bases would be difficult. Second, since the convolution result $\mathbf{s}_{oi}$ between the signal $\mathbf{s}_i$ and the kernel $\mathbf{G}$ can be expressed as

$$\begin{aligned}\mathbf{s}_{oi} &= \mathbf{V}(\hat{\mathbf{G}} \odot \hat{\mathbf{s}}_i) \\ &= \mathbf{V}\big(diag(\hat{\mathbf{G}})\hat{\mathbf{s}}_i\big) \\ &= \mathbf{V}diag(\hat{\mathbf{G}})\mathbf{V}^H\mathbf{s}_i,\end{aligned} \tag{10}$$

it is convenient to design spectrum domain $\hat{\mathbf{G}}$ directly.

Algorithm 1 Hypergraph Kernel Convolution (HKC)

Input: A point cloud of N nodes with coordinates $\mathbf{P} = [\mathbf{p}_1^T \cdots \mathbf{p}_N^T]^T$, resampling ratio α.
1. Calculate the intrinsic resolution of point cloud;
2. Use intrinsic resolution d to find coordinates $\mathbf{P_c} \in \mathbb{R}^{N_k\times3}$ of voxel centers in the kernel;
3. Use the coordinates $\mathbf{P_c}$ of voxel centers in the kernel to estimate the hypergraph spectrum basis $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_{N_k}]$ and corresponding eigenvalues $\boldsymbol{\lambda}$;
for $i = 1, 2, \cdots, N$ **do**
  4. Use hypergraph spectrum basis $\mathbf{V}$ to calculate the Fourier transform $\hat{\mathbf{s}}_i$ in (8);
  5. Calculate the Hadamard product $\hat{\mathbf{s}}_{oi} = \hat{\mathbf{s}}_i \odot \hat{\mathbf{G}}$;
  6. Calculate inverse Fourier transform $\mathbf{s}_{oi}$ using (9);
  7. Calculate local smoothness β_i in (11) ;
end for
8. Sort the local smoothness β_i in descending order and select the bottom $N_r = \alpha N$ nodes as the resampled point cloud.

In order to preserve edges in the resampled point cloud, we use a highpass filter defined in spectrum domain. Since sharp features of point clouds correspond to high frequency elements in the hypergraph spectral space [23], a highpass filter is designed to preserve edges in resampled point clouds. More specifically, in our experiments, we use a simple and well known Haar-like highpass design [7], i.e., $\hat{\mathbf{G}} = \mathbf{1} - \boldsymbol{\lambda}$, where $\boldsymbol{\lambda} = [\lambda_1\ \lambda_2 \cdots \lambda_{N_k}]^T$ are eigenvalues corresponding to eigenvectors $\{\mathbf{f}_1, \cdots, \mathbf{f}_{N_k}\}$. Since larger λ corresponds to lower frequency, Haar-like design could highlight high frequency components in the signal [19]. Note that, here we provide the Haar-like highpass filter as an example of convolution kernels. Other elegant highpass filters can also be used as the convolution kernel for the edge preserving purpose. We use the ratio between the norm of the convolution output $\mathbf{s}_{oi}$ and the norm of $\mathbf{s}_i - \mathbf{s}_{oi}$ to measure smoothness β_i for the i-th point, i.e.,

$$\beta_i = \frac{\|\mathbf{s}_{oi}\|}{\|\mathbf{s}_i - \mathbf{s}_{oi}\|}. \tag{11}$$

In resampling, we would like to extract distinct and sharp features by selecting points that exhibit lower β_i value from the resampling output. Our algorithm is summarized as Algorithm 1, also known as the HKC resampling algorithm.

2) HKC Algorithm Complexity: In an unorganized point cloud, the computational complexity of HKC amounts to $O(N^2 + N\log N + N_k(N_k+1)N + N_k^3)$, while in an organized point cloud, the complexity order is $O(N\log N + (N_k^2 + N_k + 1)N + N_k^3)$. Here are the details. First, for generating $\mathbf{s}_i$ of all points in an unorganized point cloud, the computation order is $O(N^2)$ because of the need to search the point cloud to find actual neighbors for each data point. In an organized point cloud, such computation can be reduced to $O(N)$. Next, to estimate the hypergraph spectrum basis $\mathbf{V}$ and $\mathbf{V}^H$, we need to find eigenvectors of $R_{\mathbf{P'_c}}$, requiring computation order of $O(N_k^3)$. Because the same kernel is used for every point in the point cloud, we only need to estimate hypergraph spectrum

Algorithm 2 Hypergraph Kernel Filtering (HKF)

Input: A point cloud with N nodes characterized by $\mathbf{P} = [\mathbf{p}_1^T \cdots \mathbf{p}_N^T]^T$, resampling ratio α.
1. Calculate the intrinsic resolution of point cloud;
2. Use the intrinsic resolution as d to get the coordinates $\mathbf{P_c} \in \mathbb{R}^{N_k \times 3}$ of voxel centers in the kernel;
3. Use the coordinates $\mathbf{P_c}$ of the voxel centers in the kernel to estimate the hypergraph spectrum basis $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_{N_k}]$;
for $i = 1, 2, \cdots, N$ **do**
  4. Use hypergraph spectrum basis $\mathbf{V}$ to calculate the Fourier transform $\hat{\mathbf{s}}_i$ in (8);
  5. Calculate the local smoothness σ_i in (12);
end for
6. Sort the local smoothness σ_i and select the bottom $N_r = \alpha N$ points as the resampled point cloud.

basis once. In subsequent steps, computing all HGFT and iHGFT pairs amounts to $O(NN_k^2)$. Finally, it takes $O(NN_k)$ to calculate all β_i's in (11), and additional $O(N \log N)$ to sort them.

B. Hypergraph Kernel Filtering (HKF) Resampling

To present method, we still use the $3 \times 3 \times 3$ cube as the example kernel, and the same local signal $\mathbf{s}_i$ in the Kernel convolution based method.

1) HKF Algorithm: Consider convolution via Hadamard product and inverse Fourier transform in the HKC resampling algorithm, we need to transform the local signal $\mathbf{s}_i$ from vertex domain to spectrum domain. A simpler alternative is to compute local smoothness directly for signals in spectrum domain. The computational complexity of the algorithm is reduced by eliminating the inverse transform.

For edge-preserving, we wish to separate the high frequency coefficients from the low frequency coefficients in spectrum domain. Recall that the spectrum bases corresponding to smaller λ represent higher frequency components [21]. Thus, we sort the eigenvalues of $R_{\mathbf{s}'}$ as $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{N_k}$ with corresponding eigenvectors $\{\mathbf{f}_1, \cdots, \mathbf{f}_{N_k}\}$. We can devise a threshold θ to separate the high frequency components from the low frequency components according to a sharp rise of successive eigenvalues.

Given a threshold selection of θ, we could further define a local smoothness σ_i to select the resampled points:

$$\sigma_i = \frac{\sum_{k \in \{1,2,\cdots,\theta\}} |\hat{\mathbf{s}}_i(k)|}{\sum_{k \in \{1,2,\cdots,N_k\}} |\hat{\mathbf{s}}_i(k)|}, \tag{12}$$

which is the fraction of high frequency energy within total signal energy. Finally, we resample the point cloud by selecting the points with smaller σ_i, which correspond to points containing larger amount of higher-frequency components in the hypergraph. We summarize our algorithm as Algorithm 2, also known as HKF resampling algorithm. The general steps of HKF are similar to those of HKC, while the major differences lie in the definition of smoothness as shown in Eq. (12). Since the resampled point clouds favor high-frequency points, they tend to contain more sharp features and are less smooth.

2) HKF Algorithm Complexity: Algorithm 2 (HKF) and HKC have a similar order of computational complexity, while the runtime of HKF would be shorter than HKC by eliminating the iHGFT. Similar to HKC resampling, in an unorganized point cloud, the complexity for generating $\mathbf{s}_i$ for all points is $O(N^2)$, whereas this complexity is reduced to $O(N)$ for an organized point cloud. The complexity of estimating the hypergraph spectrum basis $\mathbf{V}$ and $\mathbf{V}^H$ is $O(N_k^3)$. The cost of requisite Fourier transforms amounts to $O(NN_k^2)$. The total complexity for computing β_i in (12) is $O(NN_k)$. It further requires $O(N \log N)$ to sort the σ_i.

C. Local Hypergraph Filtering (LHF) Algorithm

In the aforementioned HKC and HKF algorithms, we use the same kernel to define all the local signals for points in the point cloud. If distribution of the points are highly non-uniform on the surface of an object, it is difficult to find a value of d suitable for every local signal. The distance parameter may either be too small for the low density part or too large for the high density part. As the result, the performance of HKC and HKF algorithms may be erratic for such highly uneven point distribution. To mitigate this problem, we further propose another HGSP approach to model the local signal by incorporating the vector between the i-th point and each of its $(N_i - 1)$ closest neighbors. In particular, we define a local signal for the i-th node of order N_i as

$$\mathbf{s}_{p,i} = [\mathbf{0}^T \ (\mathbf{p_{n_1}} - \mathbf{p_i})^T \cdots (\mathbf{p_{n_{N_i-1}}} - \mathbf{p_i})^T]^T \in \mathbb{R}^{N_i \times 3}, \tag{13}$$

where $n_1, \cdots, n_{N_i - 1}$ are the indices of its $(N_i - 1)$ nearest neighbors.

We then build hypergraphs over these points and use hyperedge to connect point i and its $(N_i - 1)$ nearest neighbors. Similar to the HKF algorithm, we also devise a filter defined in spectrum domain to process the local signal. Although, strictly speaking, we could define a unique N_i for each of the i-th point, we find it more convenient to consider some fixed selections for all points to avoid comparing hyperedges of different length.

Fig. 4 provides an example to show the effect of N_i selection. When N_i is small, $\mathbf{s}_{p,i}$ describes the local geometric information in a smaller region. Consequently, the filtered results tend to vary more and captures sharp features of the point cloud in Fig. 4a. When N_i is large, $\mathbf{s}_{p,i}$ characterize the local geometry of a larger region around the i-th point. As a result, its local information is blurred with other local information from its neighbors such that the filtered results tend to be smoother and tend to highlight the contour of the point cloud as seen from Fig. 4b.

1) Local Hypergraph Filtering (LHF) Algorithm: Because we would like to preserve both the sharp features and the surface contour of the point cloud to achieve consistently good performance across different point clouds, we propose to apply several values of N_i for all points. In particular, we consider two different lengths N_a, N_b to construct two different sets of local signals. We would then integrate the filtered results.

Our local hypergraph filtering (LHF) based resampling consists of two main steps: i) hypergraph spectrum construction,

(a) Resampled results of "dragon" using local hypergraph filtering based method with signal length $N_i = 3$. Details in the body part are kept.

(b) Resampled results of "dragon" using local hypergraph filtering based method with signal length $N_i = 6$. Details in the body part are ignored.

Fig. 4. Resampled Results of Dragon Using Local Hypergraph Filtering based Method.

ii) spectrum domain filtering. We first estimate hypergraph spectrum by applying the same process used in HKC and HKF algorithms. In this new LHF method, each point has its own (small-scale) hypergraph. We should estimate the hypergraph spectrum for each point using two different local signals $\mathbf{s}_{p,i,a} \in \mathbb{R}^{N_a \times 3}$ and $\mathbf{s}_{p,i,b} \in \mathbb{R}^{N_b \times 3}$. Once the estimation of the corresponding hypergraph spectrum bases $\mathbf{V}_{i,a}$ and $\mathbf{V}_{i,b}$ is completed, we apply (8) to derive the Fourier transform $\hat{\mathbf{s}}_{p,i,a}$ and $\hat{\mathbf{s}}_{p,i,b}$, respectively.

Similar to spectrum filter in the HKF method, we define two thresholds $\theta_{i,a}$ and $\theta_{i,b}$, respectively, for $\hat{\mathbf{s}}_{p,i,a}$ and $\hat{\mathbf{s}}_{p,i,b}$. Two local sharpness metrics are further defined as

$$\gamma(\hat{\mathbf{s}}_{p,i,a}) = \frac{\sum_{j\in\{1,2,\cdots,\theta_{i,a}\}} \sum_{k=1}^{3} |\hat{\mathbf{s}}_{p,i,a}(j,k)|}{\sum_{j\in\{1,2,\cdots,N_i\}} \sum_{k=1}^{3} |\hat{\mathbf{s}}_{p,i,a}(j,k)|}, \tag{14a}$$

$$\gamma(\hat{\mathbf{s}}_{p,i,b}) = \frac{\sum_{j\in\{1,2,\cdots,\theta_{i,b}\}} \sum_{k=1}^{3} |\hat{\mathbf{s}}_{p,i,b}(j,k)|}{\sum_{j\in\{1,2,\cdots,N_i\}} \sum_{k=1}^{3} |\hat{\mathbf{s}}_{p,i,b}(j,k)|}, \tag{14b}$$

where $\theta_{i,a}$ and $\theta_{i,b}$ correspond to the thresholds for $\hat{\mathbf{s}}_{p,i,a}$ and $\hat{\mathbf{s}}_{p,i,b}$, respectively, with N_a and N_b as the respective corresponding lengths.

Upon completion of sharpness evaluation, for each signal point, we apply a weighted average of $\gamma(\hat{\mathbf{s}}_{p,i,a})$ and $\gamma(\hat{\mathbf{s}}_{p,i,b})$ to form a combined sharpness result

$$\gamma_i = \epsilon\gamma(\hat{\mathbf{s}}_{p,i,a}) + (1-\epsilon)\gamma(\hat{\mathbf{s}}_{p,i,b}), \tag{15}$$

where ϵ denotes the weight.

To balance the effect of two local sharpness metrics, we sort both $\gamma(\hat{\mathbf{s}}_{p,i,a})$ and $\gamma(\hat{\mathbf{s}}_{p,i,b})$ and design the weight ϵ according to the top α fraction of $\gamma(\hat{\mathbf{s}}_{p,i,a})$ and $\gamma(\hat{\mathbf{s}}_{p,i,b})$, denoted by Γ_a and Γ_b, respectively. We can define node-specific weights

$$\epsilon = \frac{\Gamma_b}{\Gamma_a + \Gamma_b}. \tag{16}$$

Finally, we sort the γ_i of (15) and select the top $N_\alpha = \alpha N$ points as the resampled point cloud. The whole algorithm is summarized as Algorithm 3, also known as the LHF algorithm. In Algorithm 3, steps 1-3 correspond to the hypergraph spectrum estimation, while steps 4-6 describe the process of sharpness-based filtering as Eq. (15).

2) LHF Algorithm Complexity: The computational complexity of LHF is $O(N^2 + N\log N + N_k(N_k^2 + N_k + 1)N)$ for an unorganized point cloud, and $O(N\log N + (N_k^3 + N_k^2 + N_k + 1)N)$ for an organized point cloud. First, the complexity for generating $\mathbf{s}_{p,i}$ for all points is $O(N^2)$ and $O(N)$ for an unorganized point cloud and organized point cloud, respectively. Second, unlike HKC and HKF, the hypergraph spectrum bases would differ for each point and its corresponding local signals. Thus, one needs to estimate hypergraph spectrum basis $\mathbf{V}_{i,a}$s and $\mathbf{V}_{i,b}$s for each of point. Consequently, the total computational complexity is of $O(NN_k^3)$. Next, Fourier transforms and inverse Fourier transforms further require computation of $O(NN_k^2)$. Finally, the total complexity of computing γ_i is $O(NN_k)$, plus $O(N\log N)$ for sort the resulting γ_i.

Algorithm 3 Local Hypergraph Filtering (LHF)

Input: A point cloud with N nodes characterized by $\mathbf{P} = [\mathbf{p}_1^T \cdots \mathbf{p}_N^T]^T$, resampling ratio α, local lengths N_a, N_b.

for $i = 1, 2, \cdots, N$ **do**

1. Find the nearest $(N_a - 1)$ and $(N_b - 1)$ neighbors of point i;

2. Use coordinates of point i and its $(N_a - 1)$ and $(N_b - 1)$ neighbors to estimate the hypergraph spectrum bases $\mathbf{V}_{i,a}$, $\mathbf{V}_{i,b}$, respectively;

3. Use hypergraph spectrum basis $\mathbf{V}_{i,a}$ and $\mathbf{V}_{i,b}$ to calculate the Fourier transform $\hat{\mathbf{s}}_{p,i,a}$ and $\hat{\mathbf{s}}_{p,i,b}$, respectively;

4. Calculate the local sharpness $\gamma(\hat{\mathbf{s}}_{p,i,a})$ and $\gamma(\hat{\mathbf{s}}_{p,i,b})$ in (14a) and (14b);

end for

5. Calculate the weighted average of $\gamma_i(\hat{\mathbf{s}}_{p,i,a})$ and $\gamma_i(\hat{\mathbf{s}}_{p,i,b})$ using (15) and (16).

6. Sort the local sharpness γ_i and select the top $N_r = \alpha N$ points as the resampled point cloud.

D. Discussion

To conclude, HKC and HKF are more suitable for evenly distributed point clouds, while LHF exhibits more robust performance over unevenly distributed point clouds. All three algorithms have comparable complexity. The HKC algorithm has the complexity of $O(N^2+N\log N+N_k(N_k+1)N+N_k^3)$ for an unorganized point cloud, while in an organized point cloud, the computational complexity is only $O(N\log N + (N_k^2 + N_k + 1)N + N_k^3)$. The runtime of HKF is shorter than HKC by eliminating the iHGFT. Such difference is more significant when applying a large N_k in the convolution kernel. The runtime of LHF is higher than both HKC and HKF because of the need to estimate hypergraph spectrum basis for every local signal.

IV. Experimental results

We now describe our experiment setup and present test results of the three proposed new resampling algorithms.

A. Edge Preservation of Simple Synthetic Point Clouds

As we described in Section III, one important resampling objective is to preserve sharp features in a point cloud such as edges and corners. In this part, we study the edge preserving capability of our proposed algorithms by testing over several simple synthetic point clouds. The reason for selecting synthetic point clouds in this test is to take advantage of the known ground truth regarding edges and our ability to label them. We generate these synthetic point clouds by uniformly sampling the external surface of models constructed from combinations of cubes, cylinder and pyramid of various sizes. Examples of synthetic point clouds are shown in Fig. 5, where the points on edges are marked in red while the remaining points are in blue.

To measure the accuracy of the preserved edges, we evaluate the F_1 score, defined by

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{17}$$

where precision denotes the fraction of edge points correctly preserved among all (false or correct) edge points for a resampling algorithm while the recall is the ratio of correctly preserved edge points versus all ground truth edge points. We also calculate the mean distance to their closest ground truth edge point respectively to show the ability of the algorithms in capturing the model surface.

We compare the three proposed algorithms with graph-based resampling and traditional edge detection methods. For graph-based resampling, we use the graph-based fast resampling (GFR) method with the Haar-like highpass graph filter introduced in [7] for comparison to show the strength of hypergraph in capturing multilateral features by generalizing traditional GSP and the advantage of applying hypergraph analysis in point cloud resampling over graph based methods. In addition, we also consider an edge detection method based on eigenvalues analysis (EA) and another edge detection scheme using Principal Component Analysis (PCA) and agglomerative clustering (PCA-AC) [14]. The parameters of GFR method are

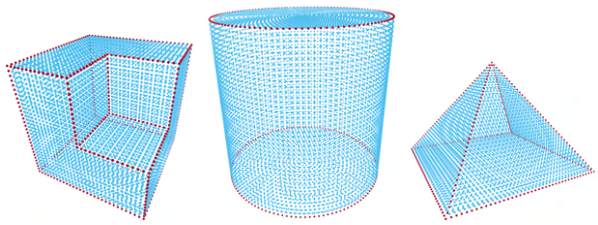


Fig. 5. Synthetic Point Clouds with Labeled Edge.

set to the typical values suggested by [7]. For EA and PCA-AC, we retain the points with higher cluster numbers or larger surface variation in the resampled point cloud to yield the same resampling ratio. Here, we set the resampling ratio $\alpha = 0.2$ for all point clouds as an example. Additional results with different resampling ratios will be further presented in Section IV-C. In order to study the robustness of algorithms, we also add 10% to 30% of Gaussian measurement noises, i.e., noises of $\mathcal{N}(0,(0.1d^{(i)})^2)$ to $\mathcal{N}(0,(0.3d^{(i)})^2)$, to point coordinates of the cloud. Here $d^{(i)}$ denotes the intrinsic resolution of point clouds. Table. I summarizes our test results. To better illustrate test results, we mark the best results among HGSP and GSP methods in bold font and underlined best results of all methods. We also mark results with different colors from warm to cold to demonstrate poor to good performance (i.e. red marks poor performance and green marks good performance).

Compared with the GSP based GFR method, the newly proposed HKC algorithm performs robustly for point clouds under larger measurement noises. Since the local signals in HKC are defined by the number of points in the voxel of kernel, weaker noises on a single point with perturbation below $d/2$ would not affect local signals. In addition, the hypergraph-based methods exhibits a smaller mean distance than GFR in all the shapes, which indicates that our proposed methods generate more robust edges than the graph-based method.

On the other hand, LHF algorithm does not perform well against noisy point clouds because sizable noises directly distort the local hypergraph and neighbors. Overall, our proposed HGSP-based methods demonstrate stronger robustness than the traditional graph-based GFR algorithm for noisy data. Using a generic signal processing approach they also deliver competitive performance against non-graph based EA and PCA-AC methods that were designed specifically for edge detection.

B. Edge Preservation Results on Real-Life Point Clouds

To test our proposed algorithm in a more general setting, we also implement edge-detection based on our resampled data in more complex and practical point clouds. For these datasets, there is no explicit ground truth edges to provide quantitative results. Therefore, we present these results as visible point cloud pictures to illustrate the test performance in Fig. 6, where the left column shows original point clouds and the right columns are the resampled point clouds for our proposed methods and methods under comparison, respectively. From Fig. 6, all methods can detect the edges of the model. Hypergraph and graph based methods tends to contain some points on a

TABLE I
NUMERICAL RESULTS OF METHODS USING POINT CLOUDS OF ALL SHAPES.

	HKC				HKF			
Noise Level	Precision	Recall	F1-Score	Mean Distance	Precision	Recall	F1-Score	Mean Distance
No Noise	0.3701	0.8558	0.4824	1.3731	0.3470	0.8581	0.4734	1.5441
10%	0.3502	0.8217	0.4579	**1.5003**	0.3265	0.7818	0.4308	1.5773
15%	**0.3344**	**0.7722**	**0.4337**	**1.6691**	0.3211	0.7480	0.4178	1.7409
20%	**0.2686**	**0.6035**	**0.3436**	**1.9078**	0.2452	0.5539	0.3116	2.0004
25%	0.2105	0.4577	0.2656	**2.3482**	**0.2153**	**0.4615**	**0.2695**	2.4392
30%	0.1774	0.3635	0.2181	**2.8434**	**0.1795**	**0.3623**	**0.2190**	2.9141
	LHF				**GFR**			
Noise Level	Precision	Recall	F1-Score	Mean Distance	Precision	Recall	F1-Score	Mean Distance
No Noise	0.3265	0.8069	0.4345	1.6662	**0.4254**	**0.9168**	**0.5324**	**1.0931**
10%	0.2017	0.5079	0.2670	1.9891	**0.3917**	**0.8903**	**0.5015**	2.7102
15%	0.1719	0.4252	0.2262	2.2813	0.3306	0.7198	0.4214	3.1772
20%	0.1492	0.3514	0.1915	2.5181	0.2407	0.5411	0.3068	3.6121
25%	0.1431	0.3339	0.1838	2.7349	0.1959	0.4182	0.2450	3.9423
30%	0.1317	0.2990	0.1677	2.9003	0.1624	0.3333	0.1994	4.1074
	EA				**PCA-AC**			
Noise Level	Precision	Recall	F1-Score	Mean Distance	Precision	Recall	F1-Score	Mean Distance
No Noise	0.3269	0.8417	0.4471	2.0033	0.3417	0.8605	0.4594	2.2471
10%	0.3264	0.8534	0.4487	1.7571	0.3451	0.8517	0.4592	1.8261
15%	0.3211	0.8429	0.4418	1.8173	0.3364	0.8379	0.4498	1.7489
20%	0.3099	0.8097	0.4251	1.9690	0.3226	0.8182	0.4349	1.7061
25%	0.2770	0.7245	0.3798	2.3033	0.3126	0.7966	0.4223	1.6027
30%	0.2377	0.6156	0.3241	2.6570	0.2945	0.7447	0.3966	1.8527

surface, while EA and PCA-AC methods tends to emphasize points closer to edges for point clouds of sofa and bookshelf in the first and second rows. We also test the algorithms on both the Boxer point cloud in 8i Voxelized Surface Light Field (8iVSLF) Dataset [32] and the Bi-plane point cloud in ScanLAB Projects [33]. We considered resampling ratio α of 0.001 and 0.005, respectively. The results are shown in Fig. 7 and Fig. 8. As shown in Fig. 7, the HKC and HKF methods can detect continuous edges and textures on the clothes of Boxer point cloud, similar to results from EA and PCA-AC. However, GFR method fails to detect continuous textures. It tends to keep more points on the entire surface. Furthermore, as shown in Fig. 8, both HKC and HKF methods are able to capture clear edges and preserve the shape of the cabin on the biplane better than other methods. We also note that the LHF method captures more clear feature of the wheels. These results show that our resampling method effectively detect 3D object contours (outlines) in real scenarios.

C. Point Cloud Recovery from Resampling

In the next test, we investigate the new algorithms' ability to preserve high degree of point cloud information after resampling. In particular, we shall attempt to recover the dense point cloud after resampling and assess the similarity between the original point cloud and the recovered point cloud from resampling.

*1) **Dense Point Cloud Recovery**:* A typical method for dense point cloud recovery consists two steps: a) reconstructing the surface of object from the resampled point cloud; and b) sampling the reconstructed object surface to generate a recovered point cloud. Since points of edge preserving resampled point clouds tend to concentrate near areas of high local variations, e.g., edges/corners, points of these resampled point clouds are not uniformly distributed, as shown in Fig. 9b. For this reason, some generic surface reconstruction methods such as Poisson reconstruction [29] may perform poorly on such sparse point clouds. We must pay special attention to surface reconstruction methods chosen for such type of resampled point cloud data.

In order to reconstruct surfaces from edge preserving and sparsely resampled point clouds, we propose to first construct the alpha complex [30] from the resampled point cloud, since this approach is well known and widely-used method for surface reconstruction based on 3D coordinates of points, and is also quite robust when handling unevenly distributed point clouds. To mitigate the potentially degrading impact of imperfect reconstruction, we decide to reconstruct six different surface models for each resampled point cloud by applying different parameters. We then apply Poisson-disk resampling to sample the alpha complex to form a recovered point cloud. To further mitigate the effect due to the possible construction of extraneous surfaces absent from the original object, we select a threshold distance d_θ three times the intrinsic resolution of the original point cloud. Using the threshold distance, we only retain the best recovered point cloud which contains the largest number of points that are within the threshold distance d_θ from the original point cloud.

*2) **Distance Between Point Clouds**:* To assess the quality of point cloud recovery, we need to define distances between the original and the recovered point clouds. Let p_i denote a point in the original and $p_{c,j}$ denote a point in the recovered point cloud. When computing our distance between two point clouds, we neglect any distances between point p_i in the original point cloud and $p_{c,j}$ in the recovered point cloud such that the minimum distances $\min_j \|p_i - p_{c,j}\|$ and $\min_i \|p_i - p_{c,j}\|$

TABLE II
EXAMPLE OF ORIGINAL AND RESAMPLED POINT CLOUDS WITH RESAMPLING RATIO $\alpha = 0.2$.

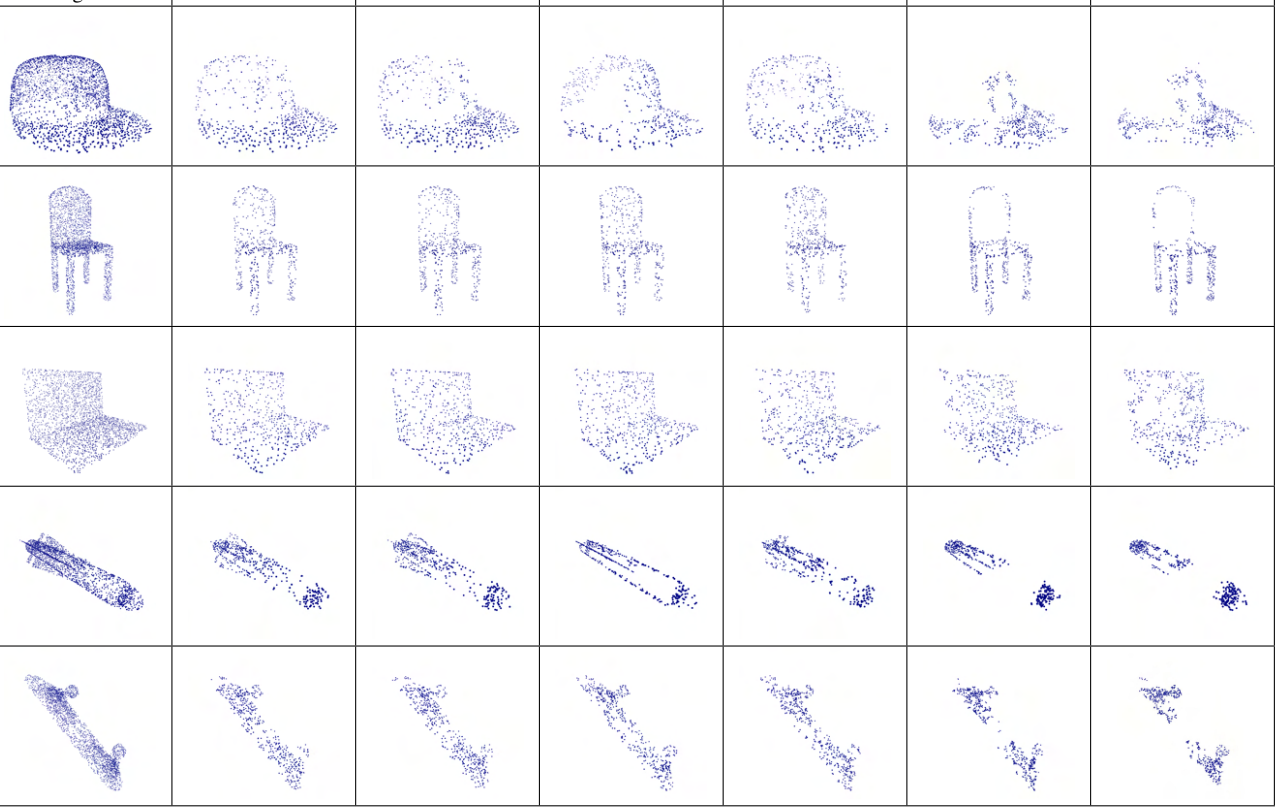
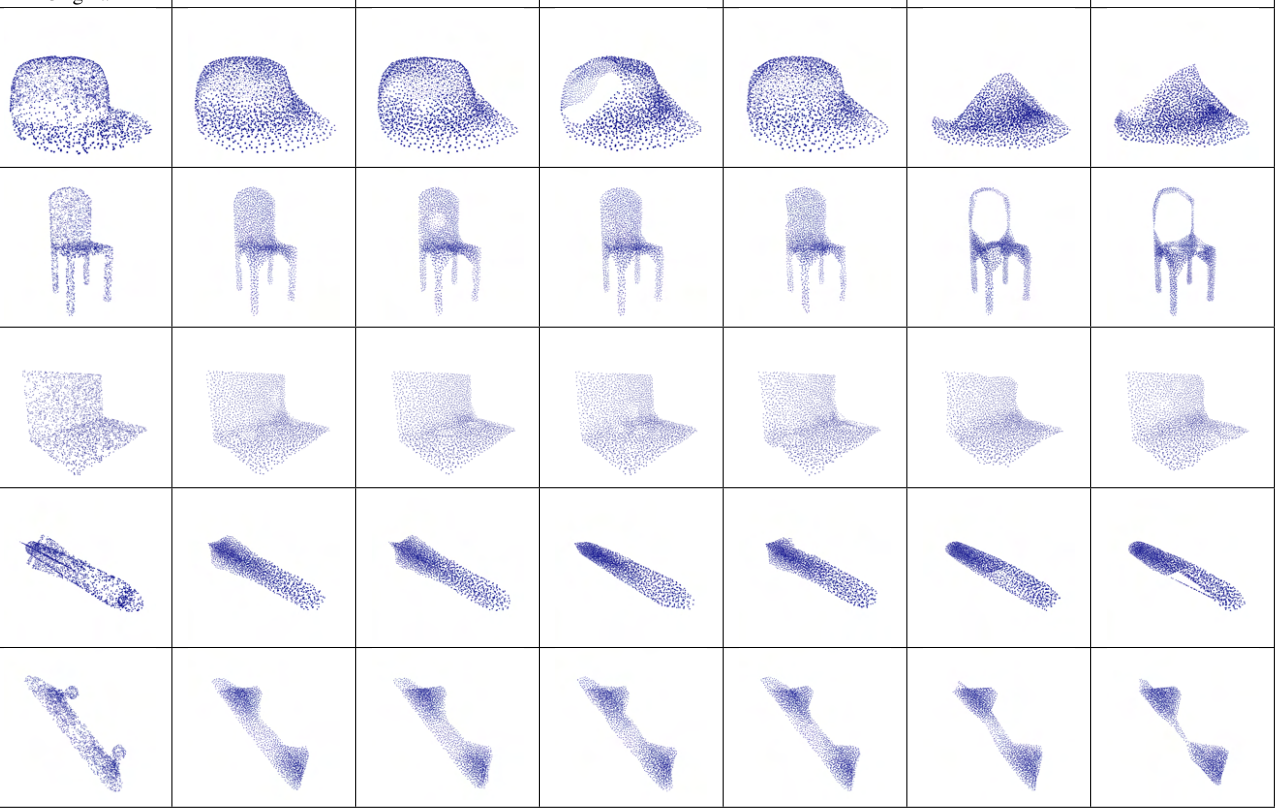
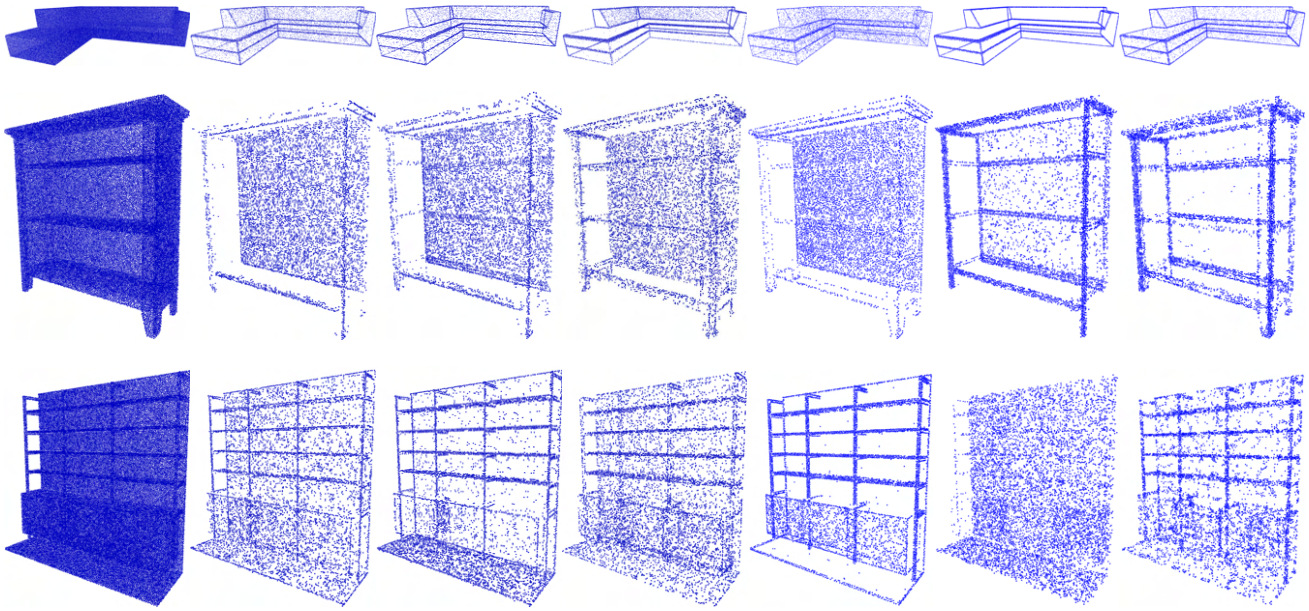
Original	HKC	HKF	LHF	GFR	EA	PCA-AC
						

TABLE III
EXAMPLE OF ORIGINAL AND RECOVERED POINT CLOUDS WITH RESAMPLING RATIO $\alpha = 0.2$.

Original	HKC	HKF	LHF	GFR	EA	PCA-AC
						



(a) Original Point Cloud (b) HKC (c) HKF (d) LHF (e) GFR (f) EA (g) PCA-AC

Fig. 6. Examples of Edge Detection for Realistic Practical Point Clouds.

(a) Original Point Cloud (b) HKC (c) HKF (d) LHF (e) GFR (f) EA (g) PCA-AC

Fig. 7. Examples of Edge Detection for Boxer Point Cloud in 8iVSLF Dataset

are greater than d_θ.

We define a distance and a dual distance between the original and the recovered point cloud as

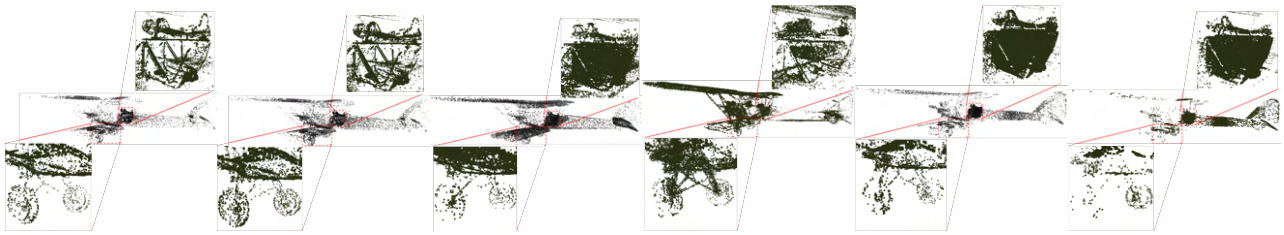
$$D_0 = \frac{1}{N_1}\sum_{i=1}^{N_1} \min_{j:\|p_i - p_{c,j}\|<d_\theta} \|p_i - p_{c,j}\|, \tag{18}$$

$$\check{D}_0 = \frac{1}{N_2}\sum_{j=1}^{N_2} \min_{i:\|p_i - p_{c,j}\|<d_\theta} \|p_i - p_{c,j}\|, \tag{19}$$

where N_1 is the number of points in the original point cloud that satisfy $\|p_i - p_{c,j}\| < d_\theta$ for some $p_{c,j}$ and N_2 is the number of points in the recovered point cloud that satisfy $\|p_i - p_{c,j}\| < d_\theta$ for some p_i. In other words, D_0 is the average distance for points that are in the original point cloud within d_θ from the closest points in the best recovered point cloud. The dual distance $\check{D}_0$ is the average distance for points in the best recovered point cloud that are within d_θ from their closest points in the original point cloud.

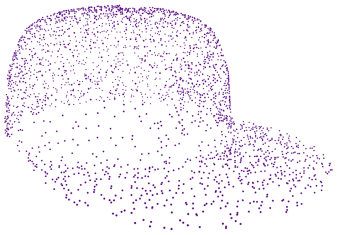
*3) **Visual and Numerical Results**:* We use six different categories of point clouds from ShapeNet [31] in our experiments. Similar to experiments discussed earlier, we test our HKF method together with the GSP-based GFR method in [7] plus the EA and PCA-AC methods from [14]. We also test each method using downsampled Boxer point cloud in 8iVSLF dataset [32]. We first uniformly downsample the Boxer point cloud with 10% points remains in the output point cloud, before sending the resulting point cloud as the input for each method. We use the downsampled point cloud for memory

(a) Original

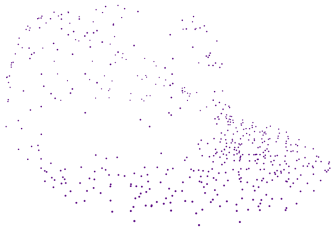


(b) HKC (c) HKF (d) LHF (e) GFR (f) EA (g) PCA-AC

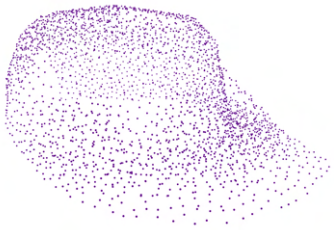
Fig. 8. Resampled results using ScanLAB Projects: Bi-plane point cloud data set



(a) Original Point Cloud of a Cap.



(b) Resampled Result with $\alpha = 0.2$.



(c) Recovered Point Cloud.

Fig. 9. Example of original point cloud, resampled point cloud and the recovered point cloud for HKC method.

and transport efficiency because the original Boxer point cloud has nearly 3,500,000 points. For fairness, we use the same resampling ratio for all the methods.

Our experiments follow the following steps. First, we apply resampling and edge detection methods to calculate the resampled point clouds with resampling ratio α. Next, we apply our proposed recovery method to generate recovered point clouds. Based on the resampled point clouds, we compute the numerical performance metrics for different algorithms in comparison. We measure the performance under three metrics: 1) distance defined in (18); 2) average of distance and dual distance as defined in (18) and (19), respectively; and 3) average number N_1 of points within the threshold d_θ between the original and recovered point cloud. Smaller distance and larger number of points within the threshold indicate better performance.

TABLE IV
MEAN DISTANCE BETWEEN THE BEST RECOVERED POINT CLOUD AND THE ORIGINAL POINT CLOUD FOR RESAMPLING RATIO $\alpha = 0.2$ USING SHAPENET DATASET.

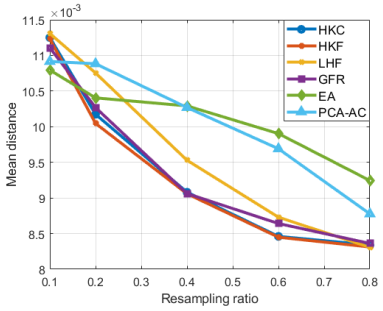
Categories	HKC	HKF	LHF	GFR	EA	PCA-AC
Cap	0.0111	0.0101	0.0117	0.0102	**0.0087**	0.0115
Chair	**0.0111**	0.0113	0.0116	0.0118	0.0125	0.0126
Laptop	0.0106	0.0106	**0.0103**	0.0105	0.0110	0.0110
Mug	**0.0134**	**0.0134**	0.0150	0.0141	0.0150	0.0147
Rocket	**0.0069**	**0.0069**	0.0078	0.0070	0.0070	0.0073
Skateboard	**0.0079**	0.0080	0.0080	**0.0079**	0.0082	0.0083
Average	0.0102	**0.0101**	0.0107	0.0103	0.0104	0.0109

TABLE V
AVERAGE NUMBER OF POINTS WITHIN d_θ BETWEEN THE BEST RECOVERED POINT CLOUD AND THE ORIGINAL POINT CLOUD FOR RESAMPLING RATIO $\alpha = 0.2$ USING SHAPENET DATASET.

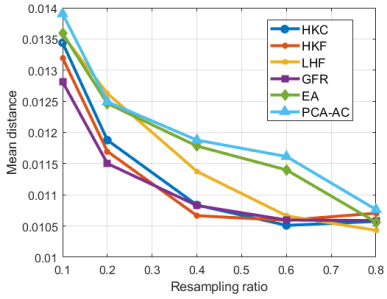
Categories	HKC	HKF	LHF	GFR	EA	PCA-AC
Cap	2628.3	2632.4	2315.7	**2635**	1427.5	2160.4
Chair	2656.1	2657.9	**2658**	2653.9	2458.2	2469.9
Laptop	2754.4	2784.4	**2785.6**	2774.3	2626.4	2584.6
Mug	2818.6	**2819.9**	2716.6	2810.7	2633.7	2418.2
Rocket	**2364.4**	2361	2317.4	2360.6	1904.4	2004.5
Skateboard	2559.8	2562	**2568.2**	2563.1	2409.9	2381.2
Average	2630.3	**2636.3**	2560.25	2632.93	2243.35	2336.47

To start, Table. II and Table. III provides a set of the resampled point clouds and their corresponding recovered ones from different methods of ShapeNet dataset. Visual inspection shows that our proposed HKC, HKF, and LHF algorithms generally deliver consistently strong results in resampling and recovery of point clouds, regardless of the dataset under study.

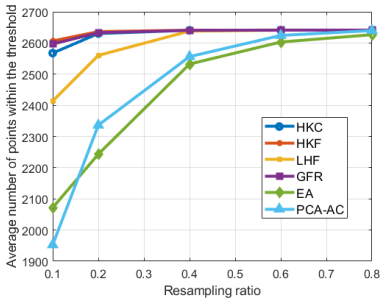
To quantitatively illustrate the performance comparison, Table. IV and Table. V presents the numerical results for $\alpha = 0.2$. From the test results, we observe that our HKF algorithm exhibits the best performance in terms of both mean distance and number of matched points versus the traditional GFR graph method and two edge detection methods. Compared with the edge detection methods, our proposed algorithms consistently retain larger numbers of points within the threshold d_θ in most point cloud categories.



(a) Mean distance against α.



(b) Average of distance and dual distance against α.

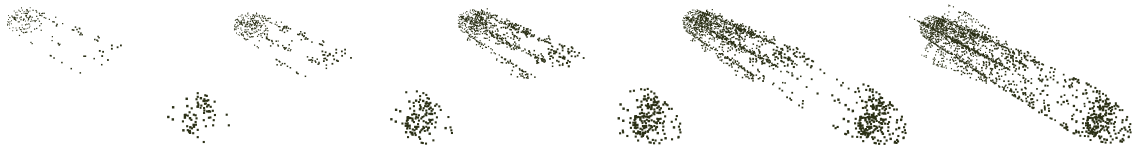


(c) Average number of points in recovered point cloud d_θ against α.

Fig. 10. Plots of recovered accuracy against resampling ratio α of all methods in ShapeNet Dataset

(a) Original Point Cloud (b) $\alpha = 0.1$ (c) $\alpha = 0.2$ (d) $\alpha = 0.4$ (e) $\alpha = 0.6$ (f) $\alpha = 0.8$

Fig. 11. Example of resampled results of EA with different α.



(a) $\alpha = 0.1$ (b) $\alpha = 0.2$ (c) $\alpha = 0.4$ (d) $\alpha = 0.6$ (e) $\alpha = 0.8$

Fig. 12. Example of resampled results of PCA-AC with different α.

TABLE VI
MEAN DISTANCE BETWEEN THE BEST RECOVERED POINT CLOUD AND THE ORIGINAL POINT CLOUD FOR DIFFERENT RESAMPLING RATIOS USING DOWNSAMPLED BOXER POINT CLOUD IN 8IVSLF DATASET.

α	**HKC**	**HKF**	**LHF**	**GFR**	**EA**	**PCA-AC**
0.1	**2.7889**	2.7942	2.6646	2.7705	2.7656	**2.3536**
0.2	2.4446	**2.1716**	2.8015	2.4362	2.7639	2.2698
0.4	1.9837	1.9841	2.4528	**1.9747**	2.7877	2.3825
0.6	1.8525	**1.8522**	1.9854	1.8522	2.8042	2.5412
0.8	1.7802	**1.7790**	1.8574	1.8598	2.8128	2.6750
Average	2.1700	**2.1162**	2.3523	2.1787	2.7868	2.4444

TABLE VII
AVERAGE OF DISTANCE AND DUAL DISTANCE BETWEEN THE BEST RECOVERED POINT CLOUD AND THE ORIGINAL POINT CLOUD FOR DIFFERENT RESAMPLING RATIOS USING DOWNSAMPLED BOXER POINT CLOUD IN 8IVSLF DATASET.

α	**HKC**	**HKF**	**LHF**	**GFR**	**EA**	**PCA-AC**
0.1	2.5368	2.5450	**2.4381**	2.4859	2.5660	4.9831
0.2	2.2974	**2.1641**	2.4492	2.3440	2.5225	2.3647
0.4	2.1475	2.0771	2.3687	**2.0614**	2.5381	5.2107
0.6	2.0890	2.0892	**2.0752**	2.0856	2.5576	5.2772
0.8	1.9825	**1.9816**	2.0204	2.0196	2.5703	2.5087
Average	2.2160	**2.1714**	2.2703	2.1993	2.5509	4.0689

The comparison also demonstrates a potential issue of the specialized edge detection based methods. In particular, resampled point cloud of edge detection methods may over-emphasize only part of the original point cloud, as shows in Fig. 11 and Fig. 12. The points in the middle to top of the rocket are kept only for α larger than or equal to 0.6. As a result, substantial number of points may not be retained by the generic edge detection methods during resampling.

We also examine the effect of sampling ratio α and graphically illustrate the variation of mean distance, average of distance and dual distance, and average number within threshold against different sampling ratios in Fig. 10, Table. VI and Table. VII. Here we use the same set of point clouds in the numerical results of ShapeNet dataset for $\alpha = 0.2$. Note that the behavior of PCA-AC is more erratic because it could over-emphasize certain part of the original point cloud, as shown in Fig. 13. It is clear and intuitive that higher resampling ratio leads to better performance of all methods under study. It is important to note that our proposed methods exhibit performances superior to the traditional EA and PCA-AC methods in terms of mean distance for various resampling ratios. This result indicates that our proposed hypergraph-based methods tend to preserve the geometric information more efficiently in resampling. The hypergraph-based methods also exhibits better results with respect to the number of corresponding nodes between reconstructed and original point clouds.

Fig. 13. Resampling result of PCA-AC method using resampled Boxer Point cloud.

D. Runtime of all methods

To compare the complexity of the proposed methods with traditional graph-based and PCA-based methods, we record the runtime of all methods using the downsampled Boxer point clouds in 8iVSLF dataset [32] with different number of points. The resampling ratio α is 0.2 for all methods and the result is as Fig. 14. We also record the average runtime of biplane point cloud with the resampling ratio of 0.005. The result is summarized in Table VIII.

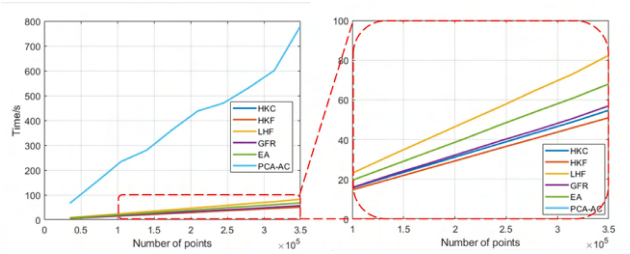


Fig. 14. Runtime of all methods

TABLE VIII
COMPARISON OF AVERAGE RUNNING TIME (IN SECONDS) FOR POINT CLOUDS IN DIFFERENT DATASETS.

	Boxer (s)	Bi-plane (s)
HKC	54.603	1501.288
HKF	50.880	1445.801
LHF	82.243	2225.681
GFR	56.815	6277.067
EA	67.840	1937.953
PCA-AC	776.976	4793.139

The runtime result shows that our HKC and HKF algorithms require shorter runtime than GFR method. The runtime of the LHF method is longer than GFR method because we used two different local signal lengths N_a and N_b such that the entire progress is calculated twice, unlike for HKC and HKF.

Note that the computational time of GFR increases drastically for the biplane point cloud. This happens because it is nearly impossible to store the entire adjacency matrix in memory and implement the GFR processing for such a large point cloud. To make the GFR method feasible, we modified GFR by calculating the optimal resampling distribution for each point directly without forming the adjacency matrix. Unlike GFR, our proposed HKC and HKF algorithms are more efficient and do not require the formation of hypergraphs for the whole point cloud.

In summary, our test results demonstrate the efficiency of the proposed resampling while preserving the underlying structural features and geometric information among point cloud data. They further demonstrate that hypergraph presents a promising alternative beyond regular graph for modeling point clouds in some point cloud related applications.

E. Parameter selection

TABLE IX
MEAN DISTANCE OF DIFFERENT METHODS.

Noise Level	No Noise	10%	15%	20%	25%	30%
HKC						
Basic (d)	1.915	2.176	3.412	4.027	4.291	4.374
Noise-dependent	↑	2.071	2.399	2.574	2.679	2.843
Optimized	1.373	1.500	1.669	1.908	2.348	2.843
HKF						
Basic (d)	1.910	2.370	3.533	4.088	4.308	4.376
Noise-dependent	↑	2.090	2.498	2.671	2.757	2.914
Optimized	1.492	1.577	1.741	2.000	2.439	2.914
LHF						
Basic (6)	3.090	3.808	4.050	4.209	4.281	4.331
Noise-dependent	↑	2.807	3.043	3.119	3.147	3.147
Optimized	1.666	1.989	2.281	2.518	2.735	2.900
GFR						
Optimized	1.093	2.710	3.177	3.612	3.942	4.107

TABLE X
ROBUSTNESS OVER DIFFERENT PARAMETERS: EDGE DETECTION FOR SYNTHETIC DATASET AND RECOVERY FOR REALISTIC DATASET.

	HKC	HKF	LHF	GFR
Variation of Mean Distance (Synthetic Dataset)	1.2321	1.1906	**0.5416**	1.4154
Variation of Average Distance (ShapeNet Dataset)	3.04E-07	**2.25E-07**	6.29E-06	6.97E-07

In this part, we provide guidelines for the parameter selection. Given a point cloud without prior knowledge of overall measurement error (noise), we suggest setting the kernel size as the resolution d and $N_i = 6$ as baseline values. When the measurement accuracy is known (i.e., the level of noise is given in practice), we recommend to increase 30% of the kernel size each time the noise grows by 10% for HKC and HKF. For LHF, we increase N_i by 4 for every 10% noise increase. We compare the mean distance of edge detection with GFR in the synthetic datasets under different guidelines as Table IX. From the results, we can see that our proposed methods deliver better performance under some simple guidelines without exhaustive tuning or hindsight.

V. CONCLUSION

This work investigates new ways for efficient and feature preserving resampling of 3D point cloud based on hypergraph signal processing (HGSP). We establish HGSP as an efficient

tool to model multilateral point relationship and to extract features in point cloud applications. We propose three new methods based on HGSP kernel convolution and spectrum filtering. Although typical HGSP tools tend to require high computational complexity, our proposed algorithms bypass certain steps for hypergraph construction and only require modest complexity to implement. Our experimental results demonstrated that the proposed hypergraph resampling algorithms can outperform traditional graph-based methods in terms of feature preservation and robustness to measurement noises.

Future works should consider the integration of HGSP methods with feature-based edge detection approaches to further enhance the edge preserving property of resampling algorithms. Another interesting direction of exploration is the design of different techniques including HGSP-based algorithms to capture multi-resolution local features and color information from point cloud data. Furthermore, we also plan to investigate HGSP extension into dynamic hypergraphs to process dynamic point clouds and videos.

REFERENCES

[1] Z. C. Marton, R. B. Rusu and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," *2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 3218-3223.

[2] R. Schnabel, S. Möser and R. Klein, "A Parallelly Decodeable Compression Scheme for Efficient Point-Cloud Rendering," in *SPBG*, Prague, Czech Republic, Sep. 2007, pp. 119-128.

[3] S. Gumhold, X. Wang and R. S. MacLeod, "Feature Extraction From Point Clouds," in *IMR*, Newport Beach, USA, Oct. 2001, pp. 293-305.

[4] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 77-85.

[5] Z. Yang, Y. Sun, S. Liu and J. Jia, "3DSSD: Point-Based 3D Single Stage Object Detector," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020, pp. 11037-11045.

[6] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler and M. Pollefeys, "Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark," *arXiv preprint arXiv:1704.03847*, Apr. 2017.

[7] S. Chen, D. Tian, C. Feng, A. Vetro and J. Kovačević, "Fast Resampling of Three-Dimensional Point Clouds via Graphs," in *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 666-681, Feb. 2018.

[8] Z. Chen, T. Zhang, J. Cao, Y. J. Zhang and C. Wang, "Point cloud resampling using centroidal Voronoi tessellation methods," *Computer-Aided Design*, vol. 102, pp. 12-21, Apr. 2018.

[9] S. Orts-Escolano, V. Morell, J. García-Rodríguez and M. Cazorla, "Point cloud data filtering and downsampling using growing neural gas," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, Aug. 2013, pp. 1-8.

[10] A. Anis, P. A. Chou and A. Ortega, "Compression of dynamic 3D point clouds using subdivisional meshes and graph wavelet transforms," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016, pp. 6360-6364.

[11] S. Chen, D. Tian, C. Feng, A. Vetro and J. Kovačević, "Contour-enhanced resampling of 3D point clouds via graphs," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, Mar. 2017, pp. 2941-2945.

[12] B. Kathariya, A. Karthik, Z. Li and R. Joshi, "Embedded binary tree for dynamic point cloud geometry compression with graph signal resampling and prediction," 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, Dec. 2017, pp. 1-4.

[13] C. Weber, S. Hahmann and H. Hagen, "Sharp feature detection in point clouds," *2010 Shape Modeling International Conference*, Aix-en-Provence, France, Jun. 2010, pp. 175-186.

[14] D. Bazazian, J. R. Casas and J. Ruiz-Hidalgo, "Fast and Robust Edge Extraction in Unorganized Point Clouds," *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Adelaide, Australia, Nov. 2015, pp. 1-8.

[15] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," in *ACM transactions on graphics (TOG)*, vol. 32, no. 1, pp. 1-12, Feb. 2013.

[16] S. Zhang, S. Cui and Z. Ding, "Hypergraph-Based Image Processing," *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 216-220.

[17] A. Bretto, (2013). Hypergraph theory. An introduction. Mathematical Engineering. Cham: Springer.

[18] A. Banerjee, A. Char and B. Mondal, "Spectra of general hypergraphs," *Linear Algebra and its Applications*, vol. 518, pp. 14-30, Apr. 2017.

[19] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura and P. Vandergheynst, "Graph Signal Processing: Overview, Challenges, and Applications," in *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808-828, May 2018.

[20] S. Barbarossa and M. Tsitsvero, "An introduction to hypergraph signal processing," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016, pp. 6425-6429.

[21] S. Zhang, Z. Ding and S. Cui, "Introducing Hypergraph Signal Processing: Theoretical Foundation and Practical Applications," in *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 639-660, Jan. 2020.

[22] S. Zhang, S. Cui and Z. Ding, "Hypergraph Spectral Clustering for Point Cloud Segmentation," in *IEEE Signal Processing Letters*, vol. 27, pp. 1655-1659, Sep. 2020.

[23] S. Zhang, S. Cui and Z. Ding, "Hypergraph Spectral Analysis and Processing in 3D Point Cloud," in *IEEE Transactions on Image Processing*, vol. 30, pp. 1193-1206, Dec. 2021.

[24] A. Afshar, J. C. Ho, B. Dilkina, I. Perros, E. B. Khalil, L. Xiong, and V. Sunderam, "Cp-ortho: an orthogonal tensor factorization framework for spatio-temporal data," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Redondo Beach, CA, USA, Jan. 2017, p. 1-4.

[25] J. Pan, and M. K. Ng, "Symmetric orthogonal approximation to symmetric tensors with applications to image reconstruction," *Numerical Linear Algebra with Applications*, vol. 25, no. 5, e2180, Apr. 2018.

[26] T. G. Kolda, "Orthogonal tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 243-255, Jul. 2006.

[27] A. K. Cherri, and M. A. Karim, "Optical symbolic substitution: edge detection using Prewitt, Sobel, and Roberts operators," *Applied Optics*, vol. 28, no. 21, pp. 4644-4648, Nov. 1989.

[28] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927-941, Nov. 2018.

[29] M. Kazhdan, M. Bolitho and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, Jun. 2006.

[30] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 1, pp. 43-72, Jan. 1994.

[31] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su and J. Xiao, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, Dec. 2015

[32] M. Krivokuća, P. A. Chou, and P. Savill, "8i Voxelized Surface Light Field (8iVSLF) Dataset," *ISO/IEC JTC1/SC29 WG11 (MPEG) input document m42914,* Ljubljana, Jul. 2018.

[33] "ScanLAB Projects: Bi-plane point cloud dataset," Available online: https://www.epfl.ch/labs/mmspg/jpeg/.

Qinwen Deng (S'20) received the B.Eng. degree in electronic information engineering from the University of Science and Technology of China, Hefei, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA, USA. His current research interests include wireless communications and hypergraph signal processing.

Songyang Zhang received the Ph.D. degree in Electrical and Computer Engineering from University of California at Davis, Davis, CA, USA. He is currently a Postdoctoral Research Associate with the Department of Electrical and Computer Engineering, University of California at Davis. His current research interests include hypergraph signal processing, behavior analysis over high dimensional graphs, and learning over graphs.

Zhi Ding (S'88-M'90-SM'95-F'03) is with the Department of Electrical and Computer Engineering at the University of California, Davis, where he holds the position of distinguished professor. He received his Ph.D. degree in Electrical Engineering from Cornell University in 1990. From 1990 to 2000, he was a faculty member of Auburn University and later, University of Iowa. Prof. Ding joined the College of Engineering at UC Davis in 2000. His major research interests and expertise cover the areas of wireless networking, communications, signal processing, multimedia, and learning. Prof. Ding supervised over 30 PhD dissertations since joining UC Davis. His research team of enthusiastic researchers works very closely with industry to solve practical problems and contributes to technological advances. His team has collaborated with researchers around the world and welcomes self-motivated young talents as new members.

Prof. Ding is a Fellow of IEEE and currently serves as the Chief Information Officer and Chief Marketing Officer of the IEEE Communications Society. He was associate editor for IEEE Transactions on Signal Processing from 1994-1997, 2001-2004, and associate editor of IEEE Signal Processing Letters 2002-2005. He was a member of technical committee on Statistical Signal and Array Processing and member of technical committee on Signal Processing for Communications (1994-2003). Dr. Ding was the General Chair of the 2016 IEEE International Conference on Acoustics, Speech, and Signal Processing and the Technical Program Chair of the 2006 IEEE Globecom. He was also an IEEE Distinguished Lecturer (Circuits and Systems Society, 2004-06, Communications Society, 2008-09). He served on as IEEE Transactions on Wireless Communications Steering Committee Member (2007-2009) and its Chair (2009-2010). Dr. Ding is a coauthor of the textbook: Modern Digital and Analog Communication Systems, 5th edition, Oxford University Press, 2019. Prof. Ding received the IEEE Communication Society's WTC Award in 2012 and the IEEE Communication Society's Education Award in 2020.