Fast Resampling of 3D Point Clouds via Graphs

Chen, S.; Tian, D.; Feng, C.; Vetro, A.; Kovacevic, J.

TR2017-215 November 2017

Abstract
To reduce the cost of storing, processing and visualizing a large-scale point cloud, we propose a randomized resampling strategy that selects a representative subset of points while preserving application-dependent features. The strategy is based on graphs, which can represent underlying surfaces and lend themselves well to efficient computation. We use a general feature-extraction operator to represent application-dependent features and propose a general reconstruction error to evaluate the quality of resampling; by minimizing the error, we obtain a general form of optimal resampling distribution. The proposed resampling distribution is guaranteed to be shift-, rotation- and scale-invariant in the 3D space. We then specify the feature-extraction operator to be a graph filter and study specific resampling strategies based on allpass, lowpass, highpass graph filtering and graph filter banks. We validate the proposed methods on three applications: large-scale visualization, accurate registration and robust shape modeling demonstrating the effectiveness and efficiency of the proposed resampling methods.

IEEE Transactions on Signal Processing

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Research Laboratories, Inc.; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Research Laboratories, Inc. All rights reserved.

Copyright © Mitsubishi Electric Research Laboratories, Inc., 2017
201 Broadway, Cambridge, Massachusetts 02139
Fast Resampling of 3D Point Clouds via Graphs

Siheng Chen, Dong Tian, Senior Member, IEEE, Chen Feng, Member, IEEE, Anthony Vetro, Fellow, IEEE, Jelena Kovačević, Fellow, IEEE

Abstract—To reduce the cost of storing, processing and visualizing a large-scale point cloud, we propose a randomized resampling strategy that selects a representative subset of points while preserving application-dependent features. The strategy is based on graphs, which can represent underlying surfaces and lend themselves well to efficient computation. We use a general feature-extraction operator to represent application-dependent features and propose a general reconstruction error to evaluate the quality of resampling; by minimizing the error, we obtain a general form of optimal resampling distribution. The proposed resampling distribution is guaranteed to be shift-, rotation- and scale-invariant in the 3D space. We then specify the feature-extraction operator to be a graph filter and study specific resampling strategies based on allpass, lowpass, highpass graph filtering and graph filter banks. We validate the proposed methods on three applications: large-scale visualization, accurate registration and robust shape modeling demonstrating the effectiveness and efficiency of the proposed resampling methods.

Index Terms—3D point clouds, sampling, graph signal processing, contour detection, visualization, registration, shape modeling

I. INTRODUCTION

With the advent of 3D sensing technologies, one can now represent objects and surrounding environments by 3D point clouds—collections of large numbers of 3D points on an object’s surface measured by a sensing device. These 3D point clouds have become popular representation tools in applications such as virtual reality, mobile mapping, scanning of historical artifacts, 3D printing and digital elevation modeling [2], among others.

A 3D point cloud is described by its 3D coordinates as well as attributes such as color, temperature and texture. Based on the storage order and spatial connectivity among 3D points, we distinguish between two types of point clouds: organized, such as those collected by a camera-like 3D sensors or 3D laser scanners and arranged on a grid, and unorganized, such as those that, due to their complex structure, are scanned from multiple view points and are subsequently merged leading to the loss of ordering of indices. Organized point clouds are easier to process as the underlying grids produce a natural spatial connectivity and reflect the sensing order. For generality, we here consider unorganized point clouds.

3D point cloud processing has become an important component in many 3D imaging and vision systems. It broadly includes compression [3], [4], [5], [6], [7], visualization [8], [9], surface reconstruction [10], [11], rendering [12], [13], editing [14], [15] and feature extraction [16], [17], [18], [19], [20]. A particular challenge is how to handle a large number of incoming 3D points [21], [22]: for example, real-time sensing systems generate millions of data points per second while in digital documentation of historical buildings and terrain visualization, we need to store billions of 3D points, making storage and subsequent processing challenging.

An approach to solving the problem is to consider efficient data structures to represent 3D point clouds. For example, [23], [24] partition the 3D space into voxels and then discretize point clouds over those voxels; to achieve fine resolution, however, a dense grid is required causing storage challenges. [25], [26] use an octree representation of point clouds, which is space efficient but suffers from discretization errors, while [27], [28] propose a probabilistic generative model to model the distribution of point clouds; such parametric models may not capture the true surface, however, and may be inefficient in terms of inferring model parameters [29], [30].

Another approach is to consider reducing the number of points through mesh simplification by constructing a triangular or polygonal mesh where nodes are 3D points (not necessarily input points) and edges are connectivities between those points respecting certain restrictions (belonging to a manifold, for example) [31], [32], [33]. The mesh is then simplified by reducing the number of nodes or edges; for example, several nodes may be merged into one with local structure preserved. Mesh construction requires costly computation, however, and mesh simplification displaces original points, causing distortion.

We here resample 3D point clouds, keeping a subset of points from the original object and reducing the number of...
3D points without changing the locations of the original ones in the process. We then design application-dependent resampling strategies to preserve application-crucial information. For example, conventional contour detection in 3D point clouds requires careful and costly computation to obtain surface normals and classification models [34], [28]. Instead, we efficiently resample a small subset of points that carries the required contour information, making the subsequent processing cheaper without losing accuracy; see Figure 1 for an example. Note that after resampling, we unavoidably lose information from the original 3D point cloud.

The proposed method is rooted in graph signal processing, a framework to explore the interaction between signals and graph structure [35], [36]. We use a graph to capture local dependencies among points, representing a discrete version of the surface of an original object. The advantage of using a graph is that one is able to capture both local and global structure of point clouds. Each of the 3D coordinates and other attributes associated with 3D points constitute a graph signal indexed by the nodes of the underlying graph. We thus formulate a resampling problem as graph signal sampling. As graph sampling methods usually select samples in a deterministic fashion necessitating solving nonconvex optimization problems to obtain samples sequentially and requiring costly computation [37], [38], [39], [40], we aim to reduce computational complexity. We thus propose an efficient randomized resampling strategy to select a subset of points; we choose subsamples according to a nonuniform resampling distribution, which provably preserves application-dependent information in the original 3D point cloud and is computational efficient (since the computational complexity scales linearly with the number of input 3D points).

- We start by proposing a feature-extraction based resampling framework. We use a general feature-extraction operator to represent application-dependent information, based on which we quantify the quality of resampling by using a simple, yet general reconstruction error, where we can derive the mean-squared error (MSE). We obtain the resampling distribution by optimizing that MSE; this distribution is guaranteed to be shift/rotation(scale-invariant).

- Next, we specify the feature-extraction operator to be a graph filter and study optimal resampling distributions based on allpass, lowpass and highpass graph filtering. In each case, we derive an optimal resampling distribution and validate the performance on both simulated and real data. The total computational cost to obtain the optimal resampling distribution scales linearly with the number of 3D points. We further combine all the proposed techniques into an efficient surface reconstruction system based on graph filter banks, which enables us to enhance features in a 3D point cloud.

- Finally, we validate these methods on three applications: (1) For large-scale visualization, we use the highpass graph filtering based resampling strategy to highlight the contours of buildings and streets in an urban scene, which avoids saturation problems in visualization. (2) For accurate registration, we use the highpass graph filtering based resampling strategy to extract the key points of a sofa, which makes the registration precise. (3) For robust shape modeling, we use the lowpass graph filtering based resampling strategy to reconstruct a surface, which makes the reconstruction robust to noise. The performances in these three applications validate the effectiveness and efficiency of the proposed resampling methods.

**Contributions.** In many large-scale 3D point cloud processing tasks including commercial software packages, resampling point clouds uniformly is widely used as a preprocessing step; despite this, this step is often approached heuristically. We instead consider 3D point cloud resampling from a novel theoretical perspective. For example, Theorem 3 shows that uniform resampling is a suboptimal resampling distribution when all 3D points are associated with the same feature value.

The main contributions of the paper are thus:

- Section III We propose a novel framework for 3D point cloud resampling. We obtain the exact MSE and optimal resampling distribution for invariant features (Theorems 1, 2) and an upper bound on the MSE and corresponding resampling distribution for variant features (Theorems 3, 4).

- Section IV We propose a novel feature-extraction operator for 3D point clouds based on graph filtering. We use Haar-like low/highpass graph filters, which is both effective and efficient (scales linearly with the number of 3D points).

- Section V We validate the proposed resampling strategies on both simulated data and real point clouds, including accurate registration, large-scale visualization and robust shape modeling.

We discuss a number of possible future directions in Section VI such as efficient 3D point cloud compression system based on graph filter banks, surface reconstruction based on arbitrary graphs and robust metric to evaluate the visualization quality of a 3D point cloud.

II. PROBLEM FORMULATION AND BACKGROUND

We now cover the background material necessary for the rest of the paper. We start by formulating the task of resampling a 3D point cloud. We then introduce graph signal processing, which lies the foundation for our proposed methods.

A. Resampling a Point Cloud

Consider a matrix representation of a point cloud with \( N \) points and \( K \) attributes,

\[
X = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \ldots & \mathbf{s}_K \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_K^T \end{bmatrix} \in \mathbb{R}^{N \times K},
\]

where \( \mathbf{s}_i \in \mathbb{R}^N \) denotes the \( i \)th attribute and \( \mathbf{x}_j \in \mathbb{R}^K \) denotes the \( j \)th point; depending on the sensing device, attributes can be 3D coordinates, RGB colors, textures, and many others. To distinguish 3D coordinates from other attributes, we store them in the first three columns of \( X \) and call that submatrix \( X_c \in \mathbb{R}^{N \times 3} \) while storing the rest of the attributes in the last \( K - 3 \) columns of \( X \) and call that submatrix \( X_o \in \mathbb{R}^{N \times (K - 3)} \).

Our task is resampling the original 3D point cloud while preserving application-crucial information by selecting \( M \) (\( M <
Let the resampled point cloud be \(X_M = \Psi X \in \mathbb{R}^{M \times K}\), where \(M = (M_1, \ldots, M_M)\) denotes the sequence of resampled indices, called resampled set, \(M_i \in \{1, \ldots, N\}\) with \(|M| = M\) and the resampling operator \(\Psi\) is a linear mapping from \(\mathbb{R}^N\) to \(\mathbb{R}^M\), defined as
\[
\Psi_{i,j} = \begin{cases} 
1, & j = M_i; \\
0, & \text{otherwise.} 
\end{cases}
\]

As the efficiency of the proposed resampling strategy is critical, we consider a randomized resampling strategy by choosing resampled indices from a resampling distribution. Let \(\pi \in \mathbb{R}^N\) be a resampling distribution, where \(\pi_i\) denotes the probability of selecting the \(i\)th sample in each random trial. Our goal is to find a resampling distribution that preserves information in the original point cloud.

The invariance properties of the proposed resampling strategy are also critical. When we shift, rotate or scale a point cloud, the intrinsic distribution of 3D points does not change and neither should the proposed resampling strategy.

**Definition 1.** A resampling strategy is called shift-invariant when, for any shift vector \(a \in \mathbb{R}^3\), input 3D point cloud \(X = [X_c, X_o]\) and corresponding resampling distribution \(\pi \in \mathbb{R}^N\), the corresponding resampling distribution to \([X_c + 1\alpha^T, X_o]\) is \(\pi\).

**Definition 2.** A resampling strategy is called rotation-invariant when, for any rotation matrix \(R \in \mathbb{R}^{3 \times 3}\), input 3D point cloud \(X = [X_c, X_o]\) and corresponding resampling distribution \(\pi \in \mathbb{R}^N\), the corresponding resampling distribution to \([X_c R, X_o]\) is \(\pi\).

**Definition 3.** A resampling strategy is called scale-invariant when, for any scaling factor \(c \in \mathbb{R}\), input 3D point cloud \(X = [X_c, X_o]\) and corresponding resampling distribution \(\pi \in \mathbb{R}^N\), the corresponding resampling distribution to \([c X_c, X_o]\) is \(\pi\).

Our aim is to guarantee that the proposed resampling strategy is shift-, rotation- and scale-invariant.

**B. Graph Signal Processing for Point Clouds**

A graph is a natural and efficient way of representing a 3D point cloud as a discretized version of an original surface. In computer graphics, a class of graphs with particular connectivity restrictions called polygon meshes are extensively used to represent the shape of an object [41]; mesh construction usually requires sophisticated geometry analysis, however, such as calculating surface normals, and the mesh representation may not be suitable for point clouds because of connectivity restrictions. We here extend the ideas behind polygon meshes to general graphs by relaxing those connectivity restrictions, gaining flexibility to capture geometry information and simplifying construction in the process.

Graph signal processing (GSP) is a theoretical framework that extends classical signal processing concepts to the analysis of high-dimensional data with complex, irregular structure [36], [42]. In particular, GSP models such data by graphs and graph signals and generalizes the concepts and tools, such as filtering and Fourier transform, from classical discrete signal processing to graph signal processing. In our work, we model all pairwise proximities between 3D points as a graph and each of 3D coordinates and attributes as a graph signal, allowing us to leverage tools from graph signal processing to deal with 3D point clouds.

**Graph Construction.** The graph consists of a set of \(N\) nodes, our input 3D points, and a set of edges among them that we construct by encoding the local geometry information through an adjacency matrix \(W \in \mathbb{R}^{N \times N}\). Let \(x_i^{(c)} \in \mathbb{R}^3\) be the 3D coordinates of the \(i\)th point (the \(i\)th row of \(X_c\)). We assign the edge weight between two points \(x_i^{(c)}\) and \(x_j^{(c)}\) as
\[
W_{i,j} = \begin{cases} 
e^{-\frac{\|x_i^{(c)} - x_j^{(c)}\|_2^2}{\sigma^2}}, & \|x_i^{(c)} - x_j^{(c)}\|_2 \leq \tau; \\
0, & \text{otherwise,} \end{cases}
\]
where variance \(\sigma\) and threshold \(\tau\) are parameters. Equation (3) shows that when the Euclidean distance of two points is smaller than a threshold \(\tau\), we connect these two points by an edge with the edge weight depending on the distance between the two points in the 3D space (the closer they are, the higher the weight). The threshold \(\tau\) controls the sparsity of the graph.

**Graph Filtering.** A graph filter is a system that takes a graph signal as an input and produces another graph signal as an output. Let \(A \in \mathbb{R}^{N \times N}\) be a graph shift operator, which is the most elementary nontrivial graph filter. Some common choices for a graph shift operator include the adjacency matrix \(W\) from (3), the transition matrix \(D^{-1} W\) (D is the weighted degree matrix, a diagonal matrix with \(D_{i,i} = \sum_j W_{i,j}\) reflecting the density around the \(i\)th point), the graph Laplacian matrix \(D - W\), and many other structure-related matrices. The graph shift replaces the signal value at a node with a weighted linear combination of values at its neighbors; that is, \(y = A s \in \mathbb{R}^N\), where \(s \in \mathbb{R}^N\) is an input graph signal (an attribute of a point cloud). Every linear, shift-invariant graph filter is a polynomial in the graph shift [45].

\[
h(A) = \sum_{\ell=0}^{L-1} h_{\ell} A^\ell = h_0 I + h_1 A + \ldots + h_{L-1} A^{L-1},
\]
where \(h_{\ell}, \ell = 0, 1, \ldots, L - 1\) are filter coefficients and \(L\) is the graph filter length. Its output is given by the matrix-vector product \(y = h(A)s\). For more details on graph filtering, see [44], [45].
Graph Fourier Transform. The eigendecomposition of a graph shift operator $A$ is \cite{46}

$$A = V \Lambda V^{-1},$$

where the eigenvectors of $A$ form the columns of matrix $V$, and the eigenvalue matrix $\Lambda \in \mathbb{R}^{N \times N}$ is the diagonal matrix of corresponding eigenvalues $\lambda_1, \ldots, \lambda_N$ of $A$. These eigenvalues are called frequencies on the graph \cite{46}. By convention, for $A = W$ or $D^{-1}W$, the eigenvalues are arranged in descending order; for $A = D-W$, the eigenvalues are arranged in ascending order. In either case, we here ensure that $\lambda_1$ is the lowest frequency and $\lambda_N$ is the highest frequency. In other words, $v_1$ captures the smallest variations in the graph and $v_N$ captures the highest variations in the graph.

In the above, $V$ is called graph Fourier basis and the graph Fourier transform of a graph signal $s \in \mathbb{R}^N$ is defined as

$$\hat{s} = V^{-1}s.$$ \hfill (6)

The inverse graph Fourier transform is $s = V\hat{s} = \sum_{k=1}^N \hat{s}_k v_k$, where $v_k$ is the $k$th column of $V$ and $\hat{s}_k$ is the $k$th component of $\hat{s}$. The vector $\hat{s}$ in (6) represents the signal's expansion in the eigenvector basis and describes the frequency components of the graph signal $s$. The inverse graph Fourier transform reconstructs the graph signal by combining graph frequency components. For more details on the graph Fourier transform, see \cite{47, 48, 49}.

Graph-based Sampling. The interest in sampling of graph signals has increased in the last few years. Sampling theory for graph signals in \cite{50, 51, 57, 52} showed perfect recovery for bandlimited graph signals based on a few subsamples; the sampling and recovery framework in \cite{53} is based on local-set-based sampling; \cite{54} proposed local-aggregated sampling; \cite{55} proposed percolation from selected seeding nodes; \cite{56} studied greedy sampling of bandlimited graph signals; \cite{57} explored the connection between sampling and sketching to handle streaming data. These sampling methods select samples in a greedy fashion and thus, the complexity does not scale linearly with the number of nodes making it impractical for handling large-scale graphs. \cite{58, 59} proposed randomized sampling strategies for approximately bandlimited graph signals; \cite{60} considered fast spectral clustering based on random sampling of bandlimited graph signals; \cite{61} showed that active sampling does not asymptotically outperform experimentally designed sampling; however, all these results are rooted in the assumption that graph signals are smooth.

Instead, here we intend to sample 3D points according to the salient features, which are not necessarily smooth over the constructed graph. Further, in terms of geometric properties of 3D point clouds, we want to ensure that the proposed resampling strategy is shift-/rotation-/scale-invariant.

III. RESAMPLING BASED ON FEATURE EXTRACTION

During resampling, the loss of information is unavoidable; our goal is thus to design an application-dependent resampling strategy that preserves application-crucial information. For example, to efficiently detect the contour in a large-scale 3D point cloud, we want to reduce the number of points while preserving the contour information. We thus select more points along the contour and fewer points outside of the contour; see Figure [1]. We consider such application-dependent information as features and call this process feature-based resampling. To preserve as many features as possible, we formulate an optimization problem for designing a resampling operator. To adapt to the geometric nature of 3D point clouds, we also guarantee that the proposed resampling strategy is shift-/rotation-/scale-invariant no matter which features are given.

Let $f(X) \in \mathbb{R}^{N \times K}$ be a feature-extraction operator that extracts application-crucial information from a point cloud\footnote{While for simplicity, we consider the number of features to be the same as the number of attributes, this is not a limitation of the method.} $X \in \mathbb{R}^{N \times K}$. Depending on the application, those features can be edges, key points and flatness \cite{17, 18, 19, 62, 20}. In this section, we consider feature-extraction operators in an abstract level and use graph filters to implement specific feature-extraction operation the next section.

We consider both invariant and variant features.

Definition 4. A feature-extraction operator $f(\cdot)$ is called shift-invariant when, for any shift vector $a \in \mathbb{R}^3$, input 3D point cloud $X = [X_c \ X_o]$ and corresponding feature $f([X_c \ X_o])$, the corresponding feature to $[X_c + 1a^T \ X_o]$ is $f([X_c \ X_o])$.

Definition 5. A feature-extraction operator $f(\cdot)$ is called rotation-invariant when, for any 3D rotation matrix $R \in \mathbb{R}^{3 \times 3}$, input 3D point cloud $X = [X_c \ X_o]$ and corresponding feature $f([X_c \ X_o])$, the corresponding feature to $[X_c R \ X_o]$ is $f([X_c \ X_o])$.

Definition 6. A feature-extraction operator $f(\cdot)$ is called scale-invariant when, for any scaling factor $c \in \mathbb{R}$, input 3D point cloud $X = [X_c \ X_o]$ and corresponding feature $f([X_c \ X_o])$, the corresponding feature to $[cX_c \ X_o]$ is $f([X_c \ X_o])$.

In what follows, we discuss resampling strategies based on each type of features.

A. Resampling Based on Invariant Features

To evaluate the performance of a resampling operator, we quantify how many features are lost during resampling; that is, we sample features, interpolate to get back the original features and compute the recovery error. The features are considered to reflect the targeted information contained in each 3D point.

We resample a point cloud $M$ times. At the $j$th step, we independently choose a point $\mathcal{M}_j = i$ with probability $\pi_i$. Let $\Psi \in \mathbb{R}^{M \times N}$ be the resampling operator from (2) and $S \in \mathbb{R}^{N \times N}$ be a diagonal rescaling matrix with $S_{ii} = 1/(M\pi_i)$. We quantify the performance of a resampling operator through

$$D_f(X)(\Psi) = \left\| S \Psi^T \Psi f(X) - f(X) \right\|_F^2,$$ \hfill (7)

where $\| \cdot \|_F$ is the Frobenius norm. $\Psi^T \Psi \in \mathbb{R}^{N \times N}$ is a zero-padding operator, a diagonal matrix with diagonal elements $(\Psi^T \Psi)_{i,i} > 0$ when the $i$th point is sampled and 0, otherwise, which ensures that the resampled and the original point clouds are of the same size. $S$ is used to compensate for nonuniform...
weights during resampling. \( S \Psi^T \) is the naive interpolation operator that reconstructs the original feature \( f(X) \) from its resampled version \( \Psi f(X) \) and \( S \Psi^T \Psi f(X) \) represents the preserved features after resampling in a zero-padded form. We consider the interpolation operator \( S \Psi^T \) for two reasons. (1) The formulation is simple and leads to a closed-form solution that will be shown in Theorems 2 and 4. (2) To ensure the robustness of the resampling operator, we want to minimize the worst-case interpolation error; this interpolation operator provides a reasonable upper bound on the interpolation error.

Note that in practice, practical interpolation operators may consider reconstructing the underlying surface, instead of original 3D points (that may not be known exactly). Because 3D points may not live on a grid, it would be hard to interpolate all points from the resampled ones without knowing the graph; however, storing the graph could be even more expensive than storing all 3D points. This is why, in practice, we prefer to reconstruct the underlying surface from resampled points and compare it with the surface reconstructed by using all 3D points. We leave this surface setting to future work.

Lemma 1 shows that \( S \) aids in providing an unbiased estimator; the proof is given in Appendix A.

**Lemma 1.** Let \( f(X) \in \mathbb{R}^{N \times K} \) be features extracted from a point cloud \( X \). Then,

\[
E_{\Psi \sim \pi} (\Psi^T \Psi f(X)) \propto \pi \odot f(X),
\]

\[
E_{\Psi \sim \pi} (S \Psi^T \Psi f(X)) = f(X),
\]

with \( E_{\Psi \sim \pi} \) the expectation over samples, which are generated from a distribution \( \pi \) independently and randomly and \( \odot \) row-wise multiplication.

The evaluation metric \( D_f(\Psi) \) measures the interpolation error, that is, how much feature information is lost during resampling, while \( E_{\Psi \sim \pi} (D_f(\Psi)) \) is the expected error caused by resampling and quantifies the performance of a resampling distribution \( \pi \).

Here \( f(\cdot) \) is shift-/rotation-/scale-invariant, \( \Psi \) does not change through shifting, rotation or scaling, leading to a shift-/rotation-/scale-invariant resampling strategy. Our goal is thus to minimize the MSE of the objective function \( (7) \).

We here consider a rotation-invariant evaluation metric,

\[
D_f(\Psi) = \max_{X'_c: \|X'_c\|_2 = c} \frac{Tr (f(X'_c) Q f(X'_c)^T)}{\|f(X'_c)\|_2},
\]

(10)

where constant \( c = \|X'_c\|_2 \) is the normalized spectral norm of 3D coordinates.

Unlike the evaluation metric \( D_f(\Psi) \) from (7), the one \( D_f(\Psi) \) from (10) removes the influence of rotation by considering the worst possible reconstruction error caused by rotation. In (10), we consider 3D coordinates as variables due to rotation. We constrain their spectral norm because it does not change during rotation (a rotation matrix is orthonormal). We then minimize \( E_{\Psi \sim \pi} (D_f(\Psi)) \) to obtain a resampling strategy that is rotation-invariant even when \( f(\cdot) \) is not.

For simplicity, we show the derivation for linear feature-extraction operators only, those of the form of \( f(X) = FX \), where \( X \) is a 3D point cloud and \( F \in \mathbb{R}^{N \times N} \) is a feature-extraction matrix; the proof is given in Appendix C.

**Theorem 3.** Let \( f(X) = FX \) be a linear, rotation-variant feature-extraction operator, with \( F \in \mathbb{R}^{N \times N} \). The MSE of the objective function (10) is

\[
E_{\Psi \sim \pi} (D_f(\Psi)) = c^2 E_{\Psi \sim \pi} \left( \sum_{k=1}^{3} \frac{\sigma_k^2}{\|f(X'_c)\|_2^2} \right) \]

\[
+ Tr \left( F X_o Q (F X_o)^T \right) \]

\[
\leq c^2 Tr \left( F Q F^T \right) + Tr \left( F X_o Q (F X_o)^T \right),
\]

(11)

(12)

The optimal resampling distribution is proportional to the magnitude of the features; that is, points with higher magnitudes have higher probability to be selected, while points with smaller magnitudes have smaller probability to be selected. The intuition is that the response after applying the feature-extraction operator reflects the information contained in each 3D point and determines the resampling probability for each.
proposed feature-extraction matrix sparse matrices that ensure low computational complexity.

Given a rotation-variant linear feature-extraction operator, it is computational expensive to minimize the exact form \( \|F\|_2 \); instead, we minimize its upper bound \( \|F\|_2 \); the proof is given in Appendix E.

**Theorem 4.** Let \( f(X) = F X \) be a linear, rotation-variant feature-extraction operator, with \( F \in \mathbb{R}^{N \times N} \). The resampling strategy \( \pi^* \) minimizing \( \|F\|_2 \) is,

\[
\pi^*_i \propto \sqrt{c^2 ||F_i||^2_2 + ||(FX_0)_i||^2_2},
\]

where constant \( c = \|X_o\|_2 \), \( F_i \) is the \( i \)-th row of \( F \) and \( (FX_0)_i \) is the \( i \)-th row of \( FX_0 \).

We see that the proposed resampling distribution is also proportional to the magnitude of features. The tuning parameter \( c \) in \( (13) \) is the normalized spectral norm used to remove the scale variance. The choice of \( c \) trades off the contribution from 3D coordinates from the other attributes.

Theorems 2 and 4 show that regardless of whether the feature-extraction operator is rotation-invariant or not, the proposed resampling operator is designed to be rotation-invariant.

### C. Computational Complexity

To obtain the rotation-variant feature-based resampling strategy \( \pi^* \), we compute the row norms of \( f(X) \); thus, the computational complexity is \( O(NK) \), which scales linearly with the number of 3D points. To obtain the rotation-variant feature-based resampling strategy \( \pi^* \), we compute the row norms of \( F \) and \( FX_0 \); thus, the computational complexity is \( O(\|\text{vec}(F)\|_0 + N)K) \), with \( \|\text{vec}(F)\|_0 \) the number of nonzero elements in \( F \). In Section 4, we show that the proposed feature-extraction matrix \( F \) is sparse, ensuring that the computational complexity still scales linearly with the number of 3D points.

#### IV. Resampling Based on Graph Filtering

The previous section studied resampling based on an arbitrary feature-extraction operator. In this section, we extract features from a point cloud by using specific graph filters \( \pi^* \), sparse matrices that ensure low computational complexity.

Let features extracted from a point cloud \( X \) be

\[
f(X) = h(A)X = \sum_{\ell=0}^{L-1} h_{\ell} A^{\ell} X.
\]

A graph filter is a linear operator; thus, the corresponding resampling distribution follows from Theorems 2 and 4 by setting \( F = \sum_{\ell=0}^{L-1} h_{\ell} A^{\ell} \). All graph filtering-based feature-extraction operators are scale-variant due to linearity, and thus, as discussed earlier, we normalize the spectral norm of the 3D coordinates. We will see that by carefully choosing the graph shift \( A \) and filter coefficients \( h_{\ell}s \), a graph filtering-based feature-extraction operator may be shift or rotation invariant.

Similarly to filter design in classical signal processing, we design a graph filter either in the graph vertex domain or in the graph spectral domain. In the graph vertex domain, for each point, a graph filter outputs a linear combination of the attributes of its neighboring points. For example, the output of the \( i \)-th point, \( f_i(X) = \sum_{\ell=0}^{L-1} h_{\ell} (A^{\ell} X)_i \), is a weighted average of the attributes of points that are within \( L \) hops away from the \( i \)-th point. The \( \ell \)-th graph filter coefficient, \( h_{\ell} \), quantifies the contribution from the \( \ell \)-th-hop neighbors. We design the filter coefficients to change the weights in local averaging.

In the graph spectral domain, we first design a desired spectral distribution and then use graph filter coefficients to fit this distribution. For example, an \( L \)-length graph filter is

\[
h(\Lambda) = V h(\Lambda) V^{-1} = V \left[ \sum_{\ell=0}^{L-1} h_{\ell} \lambda_{1}^{\ell} \cdots 0 \right] \cdots \left[ \sum_{\ell=0}^{L-1} h_{\ell} \lambda_{N}^{\ell} \right] V^{-1},
\]

where \( V \) is the graph Fourier basis and \( \lambda_i \) are graph frequencies \( (5) \). When we want the response of the \( i \)-th graph frequency to be \( c_i \), we set

\[
h(\lambda_i) = \sum_{\ell=0}^{L-1} h_{\ell} \lambda_{i}^{\ell} = c_i,
\]

and solve a set of linear equations to obtain the graph filter coefficients \( h_{\ell} \). It is also possible to use the Chebyshev polynomial to design graph filter coefficients \( 63 \). We now consider some special cases of graph filters.

#### A. Allpass Graph Filtering

Let \( h(\lambda_i) = 1 \) for all \( i \); that is, \( h(\Lambda) = I \) is an identity matrix with \( h_0 = 1 \) and \( h_{i} = 0 \) for \( i = 1, \ldots, L - 1 \). This is the setting where the original point cloud is trustworthy and all points are uniformly sampled from an object without noise, reflecting the true geometric structure of the object. We want to preserve all the information and thus, the features are the original attributes themselves. Since \( f(X) = X \), the feature-extraction operator \( f(\cdot) \) is rotation-variant. Based on Theorem 4, the corresponding resampling strategy is

\[
\pi^*_i \propto \sqrt{c^2 + ||(X_o)_i||^2_2},
\]

Here the feature-extraction matrix \( F \) is \( 9 \) an identity matrix and the norm of each row of \( F \) is 1. When we only preserve 3D coordinates and ignore other attributes, \( X_o \), we obtain a constant resampling probability for each point, meaning that uniform resampling is the resampling strategy that preserves the overall geometry information. We compute the row norms of \( X_o \) to obtain the optimal resampling strategy in \( 14 \); thus, the computational complexity is \( O(NK) \), which scales linearly with the number of 3D points.

#### B. Highpass Graph Filtering

In image processing, a highpass filter is used to extract edges and contours; similarly, we here use a highpass graph filter to extract contours in a point cloud. We only consider the 3D
coordinates as attributes \((X = X_d = \mathbb{R}^{N \times 3})\), but the proposed method can be easily extended to other attributes as well.

A critical question is how to define a contour in a 3D point cloud; some authors propose sophisticated geometry-related computations, such as surface normals, to detect contours [34]. We instead use the response of highpass graph filtering to measure local variation on graphs to detect contour points; that local variation at the \(i\)th point is

\[
f_i(X) = \| (h(A) X)_i \|_2^2,
\]

where \(h(A)\) is a highpass graph filter. The local variation \(f(X) \in \mathbb{R}^N\) quantifies the energy of the response after highpass graph filtering; when the local variation at a point is high, its 3D coordinates cannot be well approximated from those of its neighboring points, and the point is thus likely to be a contour point.

The following theorem characterizes the invariance of the local variation; the proof is given in Appendix [4]

**Theorem 5.** Let \(f(X) = \text{diag}(h(A) X X^T h(A)^T) \in \mathbb{R}^N\), where \(\text{diag}(\cdot)\) extracts the diagonal elements. \(f(X)\) is rotation invariant and shift invariant unless \(h(A) 1 = 0 \in \mathbb{R}^N\).

To guarantee shift invariance of local variation without recentering the cloud, we can use a transition matrix as a graph shift operator; that is, \(A = D^{-1} W\), where \(D\) is the diagonal degree matrix. This is because \(1 \in \mathbb{R}^N\) is the eigenvector of a transition matrix, \(A 1 = D^{-1} W 1 = 1\). Thus,

\[
h(A) 1 = \sum_{\ell=0}^{N-1} h_\ell A^\ell 1 = \sum_{\ell=0}^{N-1} h_\ell 1 = 0,
\]

when \(\sum_{\ell=0}^{N-1} h_\ell = 0\); an example is a Haar-like highpass filter

\[
h_{HH}(A) = I - A
\]

\[
= V \begin{bmatrix}
1 - \lambda_1 & 0 & \cdots & 0 \\
0 & 1 - \lambda_2 & \cdots & 0 \\
& \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1 - \lambda_N
\end{bmatrix}
V^{-1}.
\]

Note that \(\lambda_{\max} = \max_i |\lambda_i| = 1\), where \(\lambda_i\) are eigenvalues of \(A\), because the graph shift operator is a transition matrix. Then, \(h_0 = 1, h_1 = -1\) and \(h_i = 0\) for all \(i > 1\), and \(\sum_{\ell=0}^{N-1} h_\ell = 0\). Thus, a Haar-like highpass graph filter is both shift- and rotation-invariant. The graph frequency response of the filter is \(h_{HH}(\lambda_i) = 1 - \lambda_i\). Since the eigenvalues are arranged in descending order, \(1 - \lambda_i \leq 1 - \lambda_i + 1\), meaning low frequencies are attenuated and high frequencies are amplified.

In the graph vertex domain, the response of the \(i\)th point is

\[
(h_{HH}(A) X)_i = x_i - \sum_{j \in \mathcal{N}_i} A_{i,j} x_j.
\]

Because \(A\) is a transition matrix, \(\sum_{j \in \mathcal{N}_i} A_{i,j} = 1\) and \(h_{HH}(A)\) compares the difference between a point and the convex combination of its neighbors. The geometric interpretation of the proposed local variation is the Euclidean distance between the original point and the convex combination of its neighbors, reflecting how much information we can glean about a point from its neighbors. When the local variation at a point is large, the Euclidean distance between this point and the convex combination of its neighbors is large and the point provides a large amount of variation.

We validate the proposed local variation on simple examples.

**Example 1.** When a point cloud forms a line in 3D, the two endpoints belong to the contour.

**Example 2.** When a point cloud forms a polygon/polyhedron in 3D, the vertices (corner points) and the edges (line segment connecting two adjacent vertices) belong to the contour.

**Example 3.** When a point cloud forms a circle/sphere in 3D, there is no contour.

When the points are uniformly spread along the defined shape, the proposed local variation (15) satisfies Examples [1][2] and [3] from the geometric perspective. In Figure 2(a), Point 2 is the convex combination of Points 1 and 3, and the local variation of Point 2 is thus zero. However, Point 4 is not the convex combination of Points 3 and 5 and the length of the red line indicates the local variation of Point 4. Only Points 1, 4 and 7 have nonzero local variation, as expected. In Figure 2(b), all the nodes are evenly spread on a circle and have the same amount of variation, which is represented as a red line. Similar arguments show that the proposed local variation (15) satisfies Examples [1][2] and [3].

The feature-extraction operator \(f(X) = \| h_{HH}(A) X \|_F^2\) is shift and rotation-invariant. Based on Theorem 2 the optimal resampling distribution is

\[
\pi_i \propto \left\| (h_{HH}(A) X)_i \right\|_2^2 = \left\| x_i - \sum_{j \in \mathcal{N}_i} A_{i,j} x_j \right\|_2^2.
\]

where \(A = D^{-1} W\) is a transition matrix. We also recognize that \(h_{HH}(A) = I - D^{-1} W\) is the random-walk graph Laplacian matrix [64].

Note that the standard graph Laplacian matrix is commonly used to measure variation. Let \(L = D - W \in \mathbb{R}^{N \times N}\) be a graph Laplacian matrix [64]. The graph Laplacian based total variation is

\[
\text{Tr} (X^T L X) = \sum_{i,j} W_{i,j} \| x_i - x_j \|_2^2,
\]

where \(\mathcal{N}_i\) denotes the neighbors of the \(i\)th node and the variation contributed by the \(i\)th point is

\[
f_i(X) = \sum_{j \in \mathcal{N}_i} W_{i,j} \| x_i - x_j \|_2^2.
\]
Fig. 3: The pairwise difference based local variation cannot capture the contour points connecting the two faces.

The variation here is defined based on the accumulation of pairwise differences. We call (19) pairwise difference based local variation; it cannot, however, capture geometry change, for example, it violates Example 2. Figure 3 shows another example. The points are uniformly spread along the faces of a cube with each point connecting to its adjacent four points with the same edge weight. The pairwise difference based local variations of all points are the same, implying the absence of a contour; clearly, however, this is not the case (see, for example, the point to which the black arrow points in the figure).

**Computational Complexity.** To obtain the resampling strategy in (17), we compute the row norms of \( h_{HH}(A) \); the total computational complexity is thus \( O(K \| A \|_0) \), where \( \| A \|_0 \) is the number of nonzero elements in the graph shift \( A \). Because we construct a sparse graph in (3), \( \| A \|_0 \) scales linearly with the number of 3D points. Note that the proposed Haar-based highpass graph filter simply considers the first-hop neighbors with a low computational complexity; higher-order highpass graph filters could also be used at the expense of higher computational complexity.

**Experimental Validation.** Figure 4 compares the Haar-like highpass graph filtering based local variation (15) (second column) with that computed from the difference of normals (DoN) method (first column) [65], which is used to analyze point clouds for segmentation and contour detection. As a contour detection technique, DoN computes the difference between surface normals calculated at two scales. In each plot, we highlight the points that have the top 10% largest DoN scores or local variations. In Figure 4(a), we see that DoN cannot find the boundary in the plane because the surface normal does not change. The performance of DoN is also sensitive to predetermined radius. For example, the DoN cannot capture precise contours in the hinge. On the other hand, local variation captures all the contours precisely in Figure 4(b). We see similar results in Figures 4(c) and (d). Further, for each 3D point, DoN computes the first principle component of the neighboring points, a computationally inefficient procedure. The local variation only involves a sparse matrix and vector multiplication, a computationally efficient procedure.

Figure 5 shows the local variation based resampling distribution on example point clouds—hinge, cone, table, chair and trash container. The first row shows the original point clouds; the second and third rows show the resampled versions with respect to two local variations: pairwise difference based local variation (19) and Haar-like highpass graph filtering based local variation (15). The two resampled versions have the same number of points—10% of points in the original point cloud.

For the two simulated objects, the hinge and the cone (first two columns), the pairwise difference based local variation (19) fails to detect the contours while the Haar-like highpass graph filtering based local variation (15) detects all of the contours. For the real objects, the Haar-like highpass graph filtering based resampling (15) also outperforms the pairwise difference based local variation (19). In summary, the Haar-like highpass graph filtering based local variation (15) detects object contours using only 10% of points.

The highpass graph filtering based resampling strategy can be easily extended to detect transient changes in other attributes. Figure 6(a) simulates a hinge with two different textures. Black points have the same texture with value 0 and the green-circle points have a different texture with value 1. We consider texture as a new attribute in the point cloud matrix \( X \in \mathbb{R}^{N \times 4} \), where the first three columns are 3D coordinates and the fourth column is the texture. We resample 10% of points based on the highpass graph filtering based local variation (15). Figure 6(b) shows the resampled point cloud, which clearly detects both the geometric contour and the texture contour.

**C. Lowpass Graph Filtering**

In classical signal processing, a lowpass filter is used to capture a rough shape of a smooth signal and reduce noise. Similarly, we here use a lowpass graph filter to capture rough shape of a point cloud and reduce sampling noise during resampling. Since we use the 3D coordinates of points to construct a graph (3), these 3D coordinates are naturally smooth on the graph (two adjacent points in the graph have similar coordinates in the 3D space). When a 3D point cloud is corrupted by noise and outliers, a lowpass graph filter, as a denoising operator, uses local neighboring information to approximate a true position for each point. Since the output after lowpass graph filtering is a denoised version of the original point cloud, it is more appropriate to resample from denoised points than from original points.
in the lowpass graph frequency band. teapot. Figure 7(e) shows that the main energy is concentrated only ten graph frequencies, we obtain a rough structure of the teapot. 

The ideal lowpass graph filter $h_{IL}(A)$ projects an input graph signal onto a bandlimited subspace \[37\] and $b$ denotes the first $b$ columns of $V$, and the graph frequency response is

$$h_{IL}(\lambda_i) = \begin{cases} 1, & i \leq b; \\ 0, & \text{otherwise}. \end{cases}$$ (20)

The ideal lowpass graph filter $h_{IL}$ projects an input graph signal onto a bandlimited subspace \[37\] and $h_{IL}(A)s$ is a bandlimited approximation of the original graph signal $s$. We show an example in Figure 7. Figure 7(b)–(d) shows that the bandlimited approximation of the 3D coordinates of a teapot gets better as the bandwidth $b$ increases. We see that the bandwidth influences the shape of the teapot rapidly: with only ten graph frequencies, we obtain a rough structure of the teapot. Figure 7(e) shows that the main energy is concentrated in the lowpass graph frequency band.

The feature-extraction operator $f(X) = V(b)^T V(b)^T X$ is shift and rotation-varying. Based on Theorem 4, the corresponding resampling strategy is

$$\pi^*_i \propto \sqrt{c^2 \|V(b)_i\|_2^2 + \left\| \left(V(b)^T V(b) X_o \right)_i \right\|_2^2}$$ (21)

where $v_i \in \mathbb{R}^b$ is the $i$th row of $V(b)$. When we ignore other attributes $X_o$, the resampling strategy (21) is the same as the randomized resampling strategy designed for bandlimited graph signals \[58\], \[59\].

**Computational Complexity.** A direct way to obtain $\|v_i\|_2$ requires the truncated eigendecomposition \[6\], whose computational cost is $O(Nb^2)$, where $b$ is the bandwidth. It is potentially possible to approximate the leverage scores through a fast algorithm \[66\], \[67\] by using randomized techniques to avoid the eigendecomposition; the computational cost is $O(N \log(N))$. One can also partition the graph into several subgraphs and obtain leverage scores for each. Since this ideal lowpass graph filter may lead to high computational complexity, we now consider a simple, Haar-like lowpass graph filter.

2) **Haar-like Lowpass Graph Filter:** A Haar-like lowpass graph filter is

$$h_{HL}(A) = I + \frac{1}{|\lambda_{\max}|} A$$

$$= V \begin{bmatrix} 1 + \frac{\lambda_1}{|\lambda_{\max}|} & 0 & \cdots & 0 \\ 0 & 1 + \frac{\lambda_2}{|\lambda_{\max}|} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 + \frac{\lambda_{\max}}{|\lambda_{\max}|} \end{bmatrix} V^{-1},$$

where $\lambda_{\max} = \max_i |\lambda_i|$ (with $\lambda_i$ eigenvalues of $A$) helps avoid magnitude amplification. We denote $A_{\text{norm}} = A / |\lambda_{\max}|$ for simplicity. The graph frequency response is $h_{HL}(\lambda_i) = 1 + \lambda_i / |\lambda_{\max}|$. Since the eigenvalues are arranged in a descending order, we have $1 + \lambda_i \geq 1 + \lambda_{i+1}$, meaning low frequencies are amplified while high frequencies are attenuated.

In the graph vertex domain, the response of the $i$th point is $(h_{HL}(A) X)_i = x_i + \sum_{j \in N_i} (A_{\text{norm}})_{i,j} x_j$, with $N_i$ its
Corresponding to Theorem 4, the corresponding resampling strategy is

\[
\pi_i^* \propto \sqrt{c^2 \| (I + A_{\text{norm}}) X_i \|_2^2 + \|(I + A_{\text{norm}}) X_i \|_2^2}. \tag{23}
\]

Note that we can replace \( A_{\text{norm}} \) by either \( D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \) or \( D^{-1} W \); in either case, the largest eigenvalue is one, and thus \( A = A_{\text{norm}} \).

**Computational Complexity.** Similarly to the Haar-based highpass graph filter, the Haar-based lowpass graph filter also uses first-hop neighbors, with low computational cost; we could consider higher-order graph filters at the expense of more expensive computation. To obtain the resampling distribution \( (23) \), we compute the largest magnitude eigenvalue \( \lambda_{\text{max}} \) at \( O(N) \) and \( ||(I + A_{\text{norm}}) X_i ||_2 \) and \( ||(I + A_{\text{norm}}) X_{j, i} ||_2 \) for each row at \( O(\| \text{vec}(A) \|_0) \) with \( \| \text{vec}(A) \|_0 \) the nonzero elements in the \( A \), which still scales linearly with the number of 3D points.

**Experimental Validation.** We use a lowpass graph filter to handle a noisy point cloud. Figure 8(a) shows a fitness ball point cloud that contains 62,235 points collected from a Kinect device. In this noiseless case, the surface of the fitness ball can be modeled by a sphere. Figure 8(b) fits a green sphere to the fitness ball. The radius and the central point of this sphere are 0.3171 and (0.0954, 0.2077, 1.3498), respectively. To simplify computation, we resample and then fit another sphere to the resampled points. We want the spheres generated by the original point cloud and the resampled point cloud to be as close as possible.

In many real-world problems, the point cloud is noisy; to simulate it, we add Gaussian noise with mean zero and variance 0.02 to each point. Figures 9(a) and (b) show a noisy point cloud and its resampled version based on uniform resampling, respectively. Figures 9(c) and (d) show a denoised point cloud and its resampled version, respectively. Since ideal lowpass graph filtering tends to be over-smooth leading to distortion (see Figure 7), we use the Haar-like lowpass graph filtering \( (22) \) to obtain the denoised point cloud. Figure 10 compares its denoising performance with the partial differential equation (PDE)-based method [68], a standard point cloud denoising algorithm; the Haar-like lowpass graph filtering achieves a slightly better performance and is 18 times faster than the PDE-based method.

The resampling strategy can be designed based on either the Haar-like \( (22) \) or the ideal \( (21) \) lowpass graph filtering. When we set the graph shift \( A = D^{-1} W \), the corresponding Haar-like lowpass graph filtering based resampling strategy is exactly uniform sampling. Here we compare uniform sampling and the resampling strategy based on the ideal lowpass graph filtering. We fit a sphere to each of the four point clouds and show the statistics in Table 1 with relative errors, defined as 

\[
\text{Error} = \frac{|(x - \hat{x})|}{x}, \quad \text{where} \ x \ \text{is the ground truth, and} \ \hat{x} \ \text{is the estimation, in parentheses. The denoised ball (fourth column) significantly outperforms the noisy ball (third column), as the estimated radius and central point are closer to the original radius and central point. Based on the denoised ball, the ideal lowpass graph filtering based resampling (sixth column) slightly outperforms uniform sampling (fifth column). This the proposed resampling strategy as the one that provides robust shape modeling for noisy point clouds.}

\[ \]  

\section*{D. Graph Filter Banks}

In classical signal processing, a filter bank is an array of bandpass filters that break an input signal in multiple subbands and synthesize the original signal from all of the subbands [69], [70], [71]. We use a similar idea to analyze a 3D point cloud: separate an input 3D point cloud into multiple subbands and synthesize the original signal from all of the subbands [69], [70], [71]. We use a similar idea to analyze a 3D point cloud: separate an input 3D point cloud into multiple components via different resampling operators, allowing us to enhance its different components. For example, we can resample both contour points and noncontour points to reconstruct the original surfaces; we do need more contour points to emphasize contours.

Figure 11 shows a surface reconstruction system for a 3D point cloud based on graph filter banks. In the analysis part, we separate a 3D point cloud \( X \) into \( k \) subbands. In each subband, the information preserved is determined by a specific graph filter and we resample a subset of 3D points according to (9) and (13). The number of samples in each subband is determined by a sampling ratio \( \alpha \). We have flexibility to use either the original 3D points or the 3D points after graph filtering. In the synthesis part, we use the resampled points to reconstruct the surface. (A good literature review on surface reconstruction algorithms can be found in [72].) Since each surface reconstruction algorithm has its own specific set of assumptions, different surface reconstruction algorithms perform differently on the same set of 3D points.

We measure the overall performance of a surface reconstruction system by reconstruction error—the difference between the surface reconstructed from resampled points and the original surface. This leads to a rate-distortion like tradeoff: when we resample more points, we encode more bits and the reconstruction error is smaller; when we resample fewer points, we encode fewer bits and the reconstruction error is larger. The overall goal is: given a reconstruction error tolerance, we use as few samples as possible to reconstruct.
Fig. 7: Lowpass approximation represents the main shape of the original point clouds. Plot (a) shows a point cloud with 8,000 points representing a teapot. Plots (b)–(d) show the approximations with 10, 100 and 500 graph frequencies. We see that the approximation with 10 graph frequencies shows a rough structure of a teapot; the approximation with 100 graph frequencies can be recognized as a teapot; the approximation with 500 graph frequencies show some details of the teapot. Plot (e) shows the graph spectral distribution, which clearly shows that most energy is concentrated in the lowpass band.

Fig. 8: Denoising and resampling of a noisy fitness ball. Plot (b) denoises Plot (a). Plots (c) and (d) resample from Plot (b) according to uniform sampling and resampling strategy \( (21) \), respectively.

TABLE I: Proposed resampling strategy with lowpass graph filtering provides a robust shape modeling for a fitness ball. The first column is the ground truth. The relative error is shown in the parentheses. Best results are marked in bold.
The quality evaluation for 3D point clouds for future work. We know that our eyes are more sensitive to contours, a reliable way to provide quantitative results. For example, while we do not select a subset of this point cloud, run the same resampling process, and record the total elapsed time. Figure 13 shows the variation computation and resampling (in Matlab on a desktop) took less than 1,000 seconds. To test scalability, we randomly select a subset of 3D points—registration based on highpass graph filtering based resampling enhances the contour points, respectively. The registration based on highpass graph filtering based resampling precisely recovers the original point cloud, even at a detailed level. The intuition is that the highpass graph filtering based resampling enhances the contour points, which emphasizes the skeletons of both the source point cloud and the target point cloud, and reduces the interior points to remove redundant and misleading information; thus, the overall registration becomes easier. The quantitative results are shown in Table II with the root mean-squared error RMSE.

A. Contour-Based Registration

In this task, we use the proposed resampling strategy to register two point clouds efficiently and accurately. Figure 14(a) shows a point cloud of a sofa with 1,204,055 points collected from a Kinect based SLAM system. We split the original point cloud into two overlapping point clouds marked in red and blue, respectively (Figure 14). We intentionally shift and rotate the red part. The task is to invert the process and retrieve the shift and rotation. We use the iterative closest point (ICP) algorithm to register the two point clouds—a standard algorithm to rotate and shift different scans into a consistent coordinate frame. The ICP algorithm iteratively revises the rigid body transformation (combination of shift and rotation) needed to minimize the distance from the source to the reference point cloud. Figures 14(b) and (c) show the registered sofa and the details of the overlapping part after registration, respectively. We see that the registration process recovers the overall structure of the original point cloud; some mismatch is still seen at a detailed level. As it is inefficient to register two large-scale point clouds, we resample a subset of 3D points from each point cloud and then register. We compare the registration performance between a uniformly resampled point cloud and highpass graph filtering based resampled point cloud. Note that highpass graph filtering based resampling can enhance the contours and key points. Figures 14(d) and (g) show the resampled point clouds based on uniform resampling and highpass graph filtering based resampling, respectively. The two resampled versions have the same number of points (5% of points in the original point cloud). Figure 14(g) shows more contours than Figure 14(d). Based on the uniformly resampled version in Figure 14(d), Figures 14(e) and (f) show the registered sofa and the details of the overlapping part after registration, respectively. Based on the contour-enhanced resampled version Figure 14(g), Figures 14(h) and (i) show the registered sofa and the details of the overlapping part after registration, respectively. The registration based on highpass graph filtering based resampling precisely recovers the original point cloud, even at a detailed level. The intuition is that the highpass graph filtering based resampling enhances the contour points, which emphasizes the skeletons of both the source point cloud and the target point cloud, and reduces the interior points to remove redundant and misleading information; thus, the overall registration becomes easier. The qualitative results are shown in Table II with the root mean-squared error RMSE.

<table>
<thead>
<tr>
<th>Resampling Strategy</th>
<th>RMSE</th>
<th>Error_{shift}</th>
<th>Error_{rotation}</th>
</tr>
</thead>
<tbody>
<tr>
<td>All points</td>
<td>4.22</td>
<td>8.76</td>
<td>2.30 x 10^{-3}</td>
</tr>
<tr>
<td>Uniform resampling</td>
<td>4.27</td>
<td>9.38</td>
<td>3.76 x 10^{-3}</td>
</tr>
<tr>
<td>Highpass graph filtering</td>
<td>1.49</td>
<td>0.01</td>
<td>4.29 x 10^{-5}</td>
</tr>
</tbody>
</table>

TABLE II: Proposed highpass graph filtering based resampling strategy provides an accurate registration for a sofa. Best results are marked in bold. Highpass graph filtering based resampling chooses key points and provides the best registration performance.


**VI. CONCLUSIONS**

We proposed a resampling framework to extract application-dependent features from a subset of points and reduce the subsequent computation in a large-scale point cloud. We formulated an optimization problem to obtain the optimal resampling distribution, which is also guaranteed to be shift-/rotation-/scale-invariant. We then specified the feature-extraction operator to be a graph filter and studied the resampling strategies based on allpass, lowpass and highpass graph filtering. A surface reconstruction system based on graph filter banks was introduced to compress 3D point clouds. Three applications, including large-scale visualization, accurate registration and robust shape modeling were presented to validate the effectiveness and efficiency of the proposed resampling methods. We also discussed future directions, such as efficient 3D point cloud compression system based on graph filter banks, surface reconstruction based on arbitrary directions and a robust metric to evaluate the quality of a 3D point cloud.

**REFERENCES**


M. Tsitsvero, S. Barbarossa, and P. D. Lorenzo, “Signals on graphs:


APPENDIX

A. Proof of Lemma 2

Proof. Let $M$, $\Psi$ be the resampling set under the resampling distribution $\pi$ and the corresponding resampling operator, respectively. Let $\delta_{i=M_j}$ be a Bernoulli random variable indicating whether the $i$th point is selected in the $j$th selection. According to the resampling distribution $\pi$, we have

$$\delta_{i=M_j} = \begin{cases} 1, & \text{with probability } \pi_i; \\ 0, & \text{with probability } 1 - \pi_i. \end{cases}$$

According to (2), $\Psi^T \Psi$ is a diagonal, zero-padding matrix, whose $i$th element along the diagonal is

$$\left(\Psi^T \Psi\right)_{i,i} = \sum_{M_j \in M} \delta_{i=M_j},$$

which accumulates the number of times that the $i$th point is selected; in other words, $\left(\Psi^T \Psi\right)_{i,i}$ is a Binomial random variable, following $B(M, \pi_i)$. We thus have

$$E_{\psi \sim \pi} \left(\Psi^T \Psi\right)_{i,i} = M \pi_i, \quad (24)$$

$$E_{\psi \sim \pi} \left(\Psi^T \Psi\right)^2_{i,i} = M^2 \pi_i^2 + M \pi_i(1 - \pi_i). \quad (25)$$

For the nonweighted version, the $i$th row of $E_{\psi \sim \pi} \left(\Psi^T \Psi f(X)\right)$ is

$$E_{\psi \sim \pi} \left(\Psi^T \Psi f(X)\right)_{i} \overset{(a)}{=} E_{\psi \sim \pi} \left(\Psi^T \Psi\right)_{i,i} f_i(X) \overset{(b)}{=} M \pi_i f_i(X),$$

where (a) follows from that $\Psi^T \Psi$ is a diagonal matrix and (b) follows from (24). Thus, $E_{\psi \sim \pi} \left(\Psi^T \Psi f(X)\right) = M \pi \circ f(X)$, with $\circ$ the row-wise multiplication.

For the weighted version, $S \in \mathbb{R}^{N \times N}$ is a diagonal rescaling matrix, whose the $i$th element along the diagonal is $S_{i,i} = 1/(M \pi_i)$. The $i$th row of $E_{\psi \sim \pi} \left(S \Psi^T \Psi f(X)\right)$ is

$$E_{\psi \sim \pi} \left(S \Psi^T \Psi f(X)\right)_{i} = S_{i,i} E_{\psi \sim \pi} \left(\Psi^T \Psi\right)_{i,i} f_i(X) = S_{i,i} M \pi_i f_i(X) = f_i(X).$$

Thus, $E_{\psi \sim \pi} \left(S \Psi^T \Psi f(X)\right) = f(X)$. □
B. Proof of Theorem 1

Proof. The mean square error of the estimator $S \Psi^T f(X)$ is

$$
\mathbb{E}_{\Psi \sim \pi} \left[ \| S \Psi^T f(X) - f(X) \|^2_F \right]
$$

(a) $= \mathbb{E}_{\Psi \sim \pi} \left[ \text{Tr} \left( f(X)^T (S \Psi^T \Psi - I)^T (S \Psi^T \Psi - I) f(X) \right) \right]

= \text{Tr} \left( f(X)^T \mathbb{E}_{\Psi \sim \pi} \left[ (S \Psi^T \Psi - I)^T (S \Psi^T \Psi - I) \right] f(X) \right),
$$

where (a) follows from the property of the Frobenius norm. We next want to show

$$
\mathbb{E}_{\Psi \sim \pi} \left[ (S \Psi^T \Psi - I)^T (S \Psi^T \Psi - I) \right] = Q.
$$

Since both $S$ and $\Psi^T \Psi$ are diagonal matrices, $(S \Psi^T \Psi - I)^T (S \Psi^T \Psi - I)$ is also a diagonal matrix, whose element along the diagonal is

$$
\mathbb{E}_{\Psi \sim \pi} \left[ (S \Psi^T \Psi - I)^T (S \Psi^T \Psi - I) \right]_{i,i}
$$

(a) $= \mathbb{E}_{\Psi \sim \pi} \left[ (S \Psi^T \Psi)_{i,i} - 1 \right]^2

= S^2_{i,i} \mathbb{E}_{\Psi \sim \pi} \left[ (\Psi^T \Psi)_{i,i} \right] - 2 S_{i,i} \mathbb{E}_{\Psi \sim \pi} \left[ (\Psi^T \Psi)_{i,i} \right] + 1

= \frac{M^2 \pi_i^2 + M \pi_i (1 - \pi_i) - 2 M \pi_i + 1}{M^2 \pi_i}

= \frac{1 - \pi}{M \pi} = Q_{i,i},
$$

where (a) follows from the property of the diagonal matrix and (b) follows from (24) and (25). Thus, the mean square error is $\mathbb{E}_{\Psi \sim \pi} \left[ \| S \Psi^T f(X) - f(X) \|^2_F \right] = \text{Trace} \left( f(X)^T Q f(X) \right).$