Computer Aided Geometric Design 75 (2019) 101790

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Efficient L_0 resampling of point sets

Xuan Cheng [a], Ming Zeng [a,*], Jinpeng Lin [a], Zizhao Wu [b], Xinguo Liu [c]

[a] School of Informatics, Xiamen University, China
[b] School of Media and Design, Hangzhou Dianzi University, China
[c] State Key Lab of CAD&CG, Zhejiang University, China

ARTICLE INFO

Article history:
Available online 21 October 2019

Keywords:
Point set consolidation
L_0 minimization
Feature-preserving

ABSTRACT

The point data captured by laser scanners or consumer depth cameras are often contaminated with severe noises and outliers. In this paper, we propose a resampling method in an L_0 minimization framework to process such low quality data. Our framework can produce a set of clean, uniformly distributed, geometry-maintaining and feature-preserving oriented points. The L_0 norm improves the robustness to noises (outliers) and the ability to keep sharp features, but introduces a significant efficiency degradation. To further improve the efficiency of our L_0 point set resampling, we propose two accelerating algorithms including optimization-based local half-sampling and interleaved regularization. As demonstrated by the experimental results, the accelerated method is about an order of magnitude faster than the original, while achieves state-of-the-art point set consolidation performance.

1. Introduction

The emergence of laser scanners and consumer depth cameras has made it possible for rapid and convenient acquisition of point cloud from real-world objects. Due to the inherent defects of hardware and the artifacts when registering multiple point clouds, the acquired point set data are often corrupted with severe noises and outliers, which hinder the subsequent tasks such as surface reconstruction. Many point set resampling techniques have been therefore developed to deal with such imperfect data, aiming at obtaining a set of high quality consolidated points.

We think a high quality consolidated point set should satisfy five criteria: (*i*) geometry-maintaining, the resampled point set should at least "look like" the original shape; (*ii*) noise-free, the boring noises and outliers should be removed as much as possible; (*iii*) feature-preserving, the edges, corners and other sharp features should be well preserved or even enhanced; (*iv*) uniformly distribution, the resampled points are expected to be evenly distributed in space; (*v*) reliable normals, a set of reliable normals can facilitate surface reconstruction and points rendering. Although there exists many point set resampling methods, none of them can fulfill all the above five criteria completely in a scheme. The *Locally Optimal Projection* (LOP) operator (Lipman et al., 2007) is mainly designed for removing noises, which would result in local clusters if the input contains non-homogeneous point density. The variant of LOP, *Weighted LOP* (WLOP) (Huang et al., 2009), achieves more uniform distribution, but is inadequate in preserving features. *Edge-Aware Resampling* (EAR) (Huang et al., 2013b) can resample the points in a feature preserving manner, with the guidance of filtered normals. However, it is not suited to handle noises when the features are tempered with large amount of noises and outliers. Neural network based

* Corresponding author.
E-mail address: zengming@xmu.edu.cn (M. Zeng).

https://doi.org/10.1016/j.cagd.2019.101790
0167-8396/

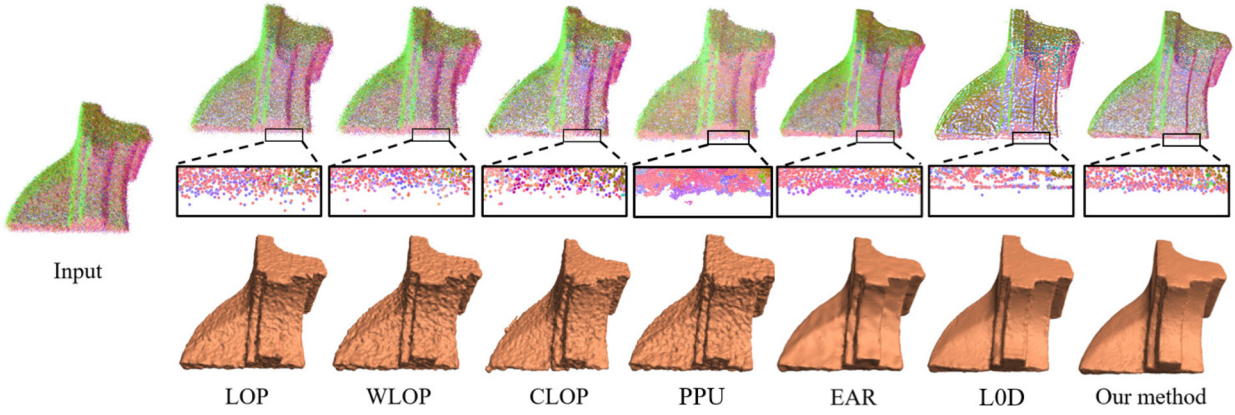


Fig. 1. Point set resampling of the "Fandisk" point set ($110K$) by our method and state-of-the-arts: LOP (Lipman et al., 2007), WLOP (Huang et al., 2009), CLOP (Preiner et al., 2014), PPU (Wang et al., 2019), EAR (Huang et al., 2013b) and L0D (Sun et al., 2015). The points are colored by their normal direction and such colorization is used throughout the paper for rendering the points. The bottom row shows the surface reconstruction results using Poisson reconstruction (Kazhdan and Hoppe, 2013). (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

resampling methods, such as *Patch-based Progressive Upsampling* (PPU) (Wang et al., 2019), can generate a high-resolution point set from a low-resolution input, but it cannot produce plausible results from particularly noisy inputs. Fig. 1 shows the performance of the above methods in a low-quality point set.

In this paper, we propose a point set resampling method in an L_0 minimization framework. In the past decade, L_0 minimization was introduced to explore the sparsity in the reconstructed signal. And it has shown great ability of noise removing and feature preserving in many computer graphics applications including image smoothing (Xu et al., 2011; Cheng et al., 2014), mesh denoising (He and Schaefer, 2013), spline approximation (Brandt et al., 2015) and volume smoothing (Wang et al., 2015). Our L_0 point set resampling method is able to generate a clean, uniform, and feature-preserving set of oriented points that well represent the underlying surface, as illustrated in Fig. 1. Specifically, the fitting to the input points guarantees a well approximation to the underlying geometry; the enforced sparsity of the non-consistency between filtered normals and optimized points, improves the resilience to noise (outliers) and the ability to preserve features; the regularization and density weights contribute to a uniform distribution in the resulting point set; the bilateral normal filtering produces a set of reliable normals.

Although L_0 norm brings many benefits, it decreases the overall processing efficiency greatly. As L_0 norm is really hard to optimize, the L_0 minimization solver (Xu et al., 2011) often takes small steps in the parameter space to approximate solutions carefully and slowly. Besides, the data size of point set is usually very large, which further increases the amount of computation. In this paper, we also provide two algorithms to improve the efficiency of our L_0 point set resampling. Firstly, we propose an optimization-based local half-sampling to reduce the number of a sample's neighbors in the input point set. Secondly, we use a interleaved regularization to cut off some repetitive computations. As demonstrated by our experimental results, the accelerated method is about an order of magnitude faster than the original, while achieves nearly the same resampling quality.

In summary, this paper offers contributions in three aspects:

- Firstly, we present an L_0 minimization based framework for point set resampling.
- Secondly, we propose two algorithms to further speed up L_0 point set resampling.
- Thirdly, we achieve state-of-the-art point set consolidation performance.

The rest of the paper is organized as follows: Section 2 introduces the related work and background, Section 3 presents our point set resampling method using L_0 minimization, Section 4 shows two accelerating algorithms, Section 5 demonstrates the experimental results and comparisons, Section 6 discusses the limitations of our proposed method, and finally Section 7 concludes this paper.

2. Related work

Point set resampling has been extensively studied over the past two decades in computer graphics, and we only review some represented works in recent years. *Moving Least Square* (MLS) (Alexa et al., 2003) has been successfully used to resample and smooth point cloud in the presence of noise. The core of MLS is to iteratively project points to a locally fit polynomial. As MLS usually assumes the underlying surface is smooth everywhere, it will have problems in handling outliers and sharp features. The *Locally Optimal Projection* (LOP) operator (Lipman et al., 2007) is driven by the L_1 median approach for data fitting, and has shown to be more effective in enhancing point sets while being robust to noises and outliers. *Weighted LOP* (WLOP) (Huang et al., 2009) deals with unevenly sampled

point clouds by taking into account a local density measure and hence reaches more evenly distributed points. *Kernel LOP* (KLOP) (Liao et al., 2013) reduces the computation cost of LOP by downsampling the original point cloud using a kernel density estimate based random sampling technique. *Continuous LOP* (CLOP) (Preiner et al., 2014) further speeds up LOP by using a Gaussian mixture model to describe the point cloud's density and then computing the attractive forces on a small set of Gaussian kernels. The *Edge-Aware Resampling* (EAR) operator (Huang et al., 2013b) is also based on LOP, but it utilizes normal information to enhance sharp features. After resampling away from edges according to the filtered normals, EAR progressively upsamples the point set to fill the edge regions. All the above methods are local operators, in which both detection and reconstruction of features are performed within a local context. This may lead to the confusion of noises and features when large amount of local noises exist.

Some global resampling methods solve for the whole point cloud at once, avoiding such defects and generating a globally optimal solution that preserves well the sharp features. The L_1-based points reconstruction method (Avron et al., 2010) formulates normals filtering and positions updating as a sequence of convex L_1 minimization problems. To further enhance sparsity, the L_0 denoising method (Sun et al., 2015) directly uses L_0 norm, reformulating normals filtering and positions updating as non-convex L_0 minimization problems. Our method is also a global method, and is closely related to the L_0 denoising (Sun et al., 2015). The L_0 denoising (Sun et al., 2015) performs optimization directly in the point position space, which may lead to a cross artifact (the updated points are more concentrated in the sharp feature regions, as shown in Fig. 1 and 7). Hence the L_0 denoising (Sun et al., 2015) requires an additional edge-recovery step to recover sharp features, and an upsampling step to get more uniformly distributed points. Our method addresses the above issues by incorporating density weights and regularization terms in the L_0 minimization. Both sharp feature preserving and uniformly distribution will be obtained in a single unified framework, without any post-processing step.

In recent years, the success of neural network techniques in image space encourages the development of resampling methods for 3D point cloud. Neural point cloud processing is pioneered by PointNet (Qi et al., 2017a) and its multiscale variant PointNet++ (Qi et al., 2017b). Through applying shared multilayer perceptrons for the feature transformation of individual points, the problem of irregularity and the lack of structure is addressed. Later, PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) are successfully applied in point cloud consolidation and upsampling. Zhang et al. (2018) trains a PointNet-based point generation model on the entire object for point cloud upsampling, but its application is limited to low-resolution input. PU-net (Yu et al., 2018a) and EC-net (Yu et al., 2018b) operate on patch level, and thus can handle high-resolution input. The most recent neural point cloud upsampling work is PPU (Wang et al., 2019), which uses a multi-step, patch-based architecture to channel the attention of the network to both global and local features. However, the loss function used in the training process of PPU (Wang et al., 2019) is mainly designed for keeping close to the ground truth, without consideration of noises removing and feature preserving. We have made comprehensive comparisons between our method and PPU (Wang et al., 2019) in the experiments.

Surface reconstruction generates a triangle mesh model from a set of points. In many applications, point set resampling usually acts as a pre-processing step for surface reconstruction. Existing surface reconstruction methods can be classified into combinatorial methods and implicit methods. The well-known combinatorial methods are Cocone (Dey and Giesen, 2001) and its various extensions (Amenta et al., 2002; Dey and Wang, 2013), which attempt to find a piecewise linear approximation of input point set using Delaunay triangles (Kolluri et al., 2004). The implicit methods construct implicit functions for the underlying surface, including locally fitting tangent planes (Hoppe et al., 1992), using Radial Basis Functions (Carr et al., 2001) and Poisson Reconstruction (Kazhdan et al., 2006; Kazhdan and Hoppe, 2013). There exists a public benchmark (Berger et al., 2013) for comprehensive evaluation and comparison of surface reconstruction methods.

L_0 **minimization** is firstly introduced to do edge-preserving image smoothing (Xu et al., 2011) in computer graphics. Due to the non-convex and non-derivative nature of L_0 norm, it is really hard to solve the L_0 minimization problem. In (Xu et al., 2011), an alternative minimization algorithm was proposed to approximate the solution, in which a set of auxiliary variables are introduced and the original non-convex problem is decomposed to a sequence of computationally tractable L_0-L_2 problems. Then the fused coordinate descent algorithm (Cheng et al., 2014) was proposed to get more sparse solutions. The spirit of this algorithm is that only one variable is optimized at a time, and the neighboring variables are fused together once their values are equal. Recently, a modified random coordinate descent algorithm (Brandt et al., 2015) was designed. In each iteration, it randomly selects a index of vertices to be constrained, and then solves a linearly constrained quadratic program. We choose the alternative minimization algorithm (Xu et al., 2011) as our L_0 solver, because it is the most frequently used solver currently.

3. L_0 point set resampling

The input to our method is a set of unorganized points $Q = \{q_j\}_{j\in J} \subset \mathbb{R}^3$, typically unevenly distributed, without normals and containing noises and outliers. The output of our method is an oriented point set $S = \{(p_i, \mathbf{n}_i)\}_{i\in I} \subset \mathbb{R}^6$ consisting of cleaned point positions p_i that represent well the underlying piece-wise smooth surface, and their associated reliable normals $\mathbf{n}_i$.

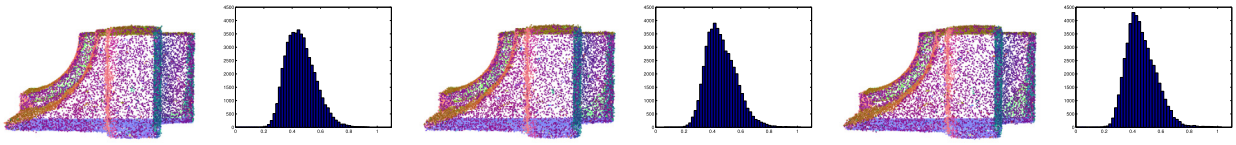


Fig. 2. (a), (c) and (e) show the resampling points by Equation (3), (4) and (6). (b), (d) and (f) show the points' local density distribution of (a), (c) and (e). The X-axis represents the point's local density computed by Equation (5), while the Y-axis represents the frequency.

3.1. Normal filtering

Starting with the input point set Q, we use the popular PCA-based method (Hoppe et al., 1992) to compute the surface normals. The resulting PCA normals can be unreliable especially near the sharp feature regions, when the input contains high noise or is non-uniformly distributed. Since the subsequent resampling process highly relies on a set of reliable and feature-preserving normals, we use the bilateral normal filtering (Öztireli et al., 2009; Huang et al., 2013b) to re-estimates the normals from anisotropic neighborhood. Specifically, for a given point q_j in Q associated with its PCA normal $\mathbf{n}_j$, we define its neighborhood as $\mathcal{N}_{q_j} = \{q_{j'} | q_{j'} \in Q \wedge ||q_j - q_{j'}|| < r_q\}$. $||.||$ is the L_2 norm and r_q is the neighborhood size. Then we iteratively update $\mathbf{n}_j$ for all j with

$$\mathbf{n}_j = \frac{\sum_{q_{j'} \in \mathcal{N}_{q_j}} \theta(||q_j - q_{j'}||)\psi(\mathbf{n}_j, \mathbf{n}_{j'})\mathbf{n}_{j'}}{\sum_{q_{j'} \in \mathcal{N}_{q_j}} \theta(||q_j - q_{j'}||)\psi(\mathbf{n}_j, \mathbf{n}_{j'})}, \tag{1}$$

where the spatial weight function $\theta(r)$ and normal weight function $\psi(\mathbf{n}_j, \mathbf{n}_{j'})$ are defined as

$$\theta(d) = e^{-d^2/{\sigma_q}^2}, \quad \psi(\mathbf{n}_j, \mathbf{n}_{j'}) = e^{-(\frac{1-\mathbf{n}_j{}^{\mathrm{T}}\mathbf{n}_{j'}}{1-cos(\sigma_n)})^2}. \tag{2}$$

σ_q is the position parameter, and is typically set as a scaling of the neighborhood size r_q. σ_n is the angle parameter that measures the similarity of neighboring normals, and we set $\sigma_n = 15°$ defaultly. The bilateral normal filtering averages the normals in the neighborhood with weights accounting for both position difference and normal difference. Due to the high variance PCA normals in the vicinity, the above formula works well to distinguish normals across discontinuities, classifying their directions into two disjoint clusters near each sharp edge. In this way, the sharp feature information is supposed to be well encoded in the filtered normals.

3.2. L_0 resampling

Basic formulation. With the high-quality filtered normals, we now describe how to formulate the point set resampling in an L_0 minimization framework. We would like to minimize the sum of squared distances of the resampled points S to their local neighborhood in Q. Furthermore, the resampled points S should be consistent with their filtered normals, under a local planarity assumption. Hence, the formulation of L_0 resampling is given by:

$$\min_{P=\{p_i\}} \left\{ \sum_{i \in I} \sum_{q_j \in \mathcal{M}_{p_i}} ||p_i - q_j||^2 + \lambda \sum_{i \in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} ||\mathbf{n}_i{}^{\mathrm{T}}(p_i - p_{i'})||_0 \right\}. \tag{3}$$

$||.||_0$ is the L_0 norm which counts the number of non-zero elements in a vector. $\mathcal{M}_{p_i} = \{q_j | q_j \in Q \wedge ||p_i - q_j|| < r_p\}$ is defined as the neighborhood of p_i in the input point set Q. $\mathcal{N}_{p_i} = \{p_{i'} | p_{i'} \in S \wedge ||p_{i'} - p_i|| < \frac{r_p}{2}\}$ is defined as the neighborhood of p_i in the resulting point set S. For convenience, we refer $\mathcal{M}_{p_i}$ as *original neighbors*, and $\mathcal{N}_{p_i}$ as *self neighbors* throughout this paper. The radius parameter r_p is adapted to the input point set Q. It is typically set as $r_p = l_{diag}/\sqrt[3]{|Q|}$, where l_{diag} is the diagonal length of the bounding box of Q, and $|Q|$ is the number of points in Q. The parameter λ is used to effectively control the degree of consistency between the optimized point set $\{p_i\}$ and the filtered normals $\{\mathbf{n}_i\}$. λ can be adjusted by the user, with a large value resulting in more sharper edges and corners while a small value maintaining the original shape as much as possible.

Regularization. The solution of Equation (3) produces a good approximation to the input surface and a clean preservation of the sharp features. However, in many cases, the resulting points have an irregular spatial distribution and tend to accumulate in local clusters. We show an example in Fig. 2(a). Our solution is to add a self neighbor regularizer for each neighboring pair:

$$\min_{P} \left\{ \sum_{i \in I} \sum_{q_j \in \mathcal{M}_{p_i}} ||p_i - q_j||^2 + \alpha \sum_{i \in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} ||p_i - p_{i'}||^2 \bar{\theta}(||p_i - p_{i'}||) + \lambda \sum_{i \in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} ||\mathbf{n}_i{}^{\mathrm{T}}(p_i - p_{i'})||_0 \right\}. \tag{4}$$

α is weight of the regularization term. $\bar{\theta}(d) = e^{d^2/{r_p}^2}$ is the spatial weight function, which has a different form with Equation (2). The regularization term will penalize the neighboring pairs with large spatial distance more greatly, and thus encourage them to move closer. On the other hand, the regularization term has minor effects on the pairs that are already close enough. With the interactions of all neighboring pairs, we get a more uniformly distributed result than Equation (3).

Density weights. Adding the regularization term, the Equation (4) works well for the input point set Q with relatively uniform spatial distribution. The data fit term in the above optimization is closely related to the L_1 median, which leads to projected points moving toward the local distribution center of the input point set Q. If Q is highly non-uniform, the solution of Equation (4) tends to follow the trend of such non-uniformity. In this case, the regularization term has a relatively limited function, no matter how high the weight value α is. Inspired by WLOP (Huang et al., 2009), we incorporate two types of locally adaptive density weights to Equation (4). For each point q_j in Q and p_i in S, the local density weights are given:

$$\begin{aligned} v_j &= 1 + \sum_{q_{j'} \in \mathcal{N}_{q_j}} \theta(||q_j - q_{j'}||)/\eta_Q\,, \\ w_i &= 1 + \sum_{p_{i'} \in \mathcal{N}_{p_i}} \theta(||p_i - p_{i'}||)/\eta_S, \end{aligned} \tag{5}$$

where η_Q is the average neighbors' number in Q and η_S is the average neighbors' number in S. The measure of the density of a point takes two elements into consideration. Firstly, if a point has more neighbors under a certain radius, there would be denser around this point. Secondly, if the neighbors are more close to this point, the possibility of being dense would be higher. Taking the local density weights $\{v_j\}$ and $\{w_i\}$ into Equation (4), we arrive at the final version of our L_0 point set resampling:

$$\min_P \left\{ \sum_{i\in I} \sum_{q_j \in \mathcal{M}_{p_i}} ||p_i - q_j||^2/v_j + \alpha \sum_{i\in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} ||p_i - p_{i'}||^2 \bar{\theta}(||p_i - p_{i'}||)/w_i + \lambda \sum_{i\in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} ||{\mathbf{n}_i}^{\mathrm{T}}(p_i - p_{i'})||_0 \right\}. \tag{6}$$

Thus, the attraction of point clusters in the given set Q is relaxed by the local density weight v_j. For example, if the region around the point q_j is very dense in Q, we don't expect the projected point p_i to approach it. With the same spirit, the regularization forces are also relaxed by w_i.

The effectiveness of both the regularization and the density weights are verified in Fig. 2, where the results by Equation (3), (4) and (6) are presented. It could be observed that the resulting points by Equation (6) have the most centralized density distribution (Fig. 2(f)), and thus enjoy the most uniform spatial distribution (Fig. 2(e)).

3.3. Alternative minimization solver

The main strategy of the alternative minimization algorithm (Xu et al., 2011) is to introduce a set of auxiliary variables, and then alternatively solve a sequence of linear quadratic problems. Specifically, through introducing the auxiliary variables $\{\delta_{i,i'}\}$, the Equation (6) becomes:

$$\begin{aligned} \min_{P,\,\delta} \Bigg\{ & \sum_{i\in I} \sum_{q_j \in \mathcal{M}_{p_i}} ||p_i - q_j||^2/v_j + \alpha \sum_{i\in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} ||p_i - p_{i'}||^2 \bar{\theta}(||p_i - p_{i'}||)/w_i \\ & + \beta \sum_{i\in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} (||{\mathbf{n}_i}^{\mathrm{T}}(p_i - p_{i'}) - \delta_{i,i'}||^2 + \lambda ||\delta_{i,i'}||_0) \Bigg\}. \end{aligned} \tag{7}$$

We optimize this expression with the alternating optimization. Firstly, δ is optimized with P fixed:

$$\min_{\delta} \sum_{i\in I} \sum_{p_{i'} \in \mathcal{N}_{p_i}} (\beta ||{\mathbf{n}_i}^{\mathrm{T}}(p_i - p_{i'}) - \delta_{i,i'}||^2 + \lambda ||\delta_{i,i'}||_0). \tag{8}$$

In this minimization, each entry $\delta_{i,i'}$ is independent with each other, and can be solved efficiently by using the shrinkage thresholding operator with the following closed form:

$$\delta_{i,i'} = \begin{cases} 0, & \text{if } {\mathbf{n}_i}^{\mathrm{T}}(p_i - p_{i'}) < \sqrt{\lambda/\beta}. \\ {\mathbf{n}_i}^{\mathrm{T}}(p_i - p_{i'}), & \text{otherwise.} \end{cases} \tag{9}$$

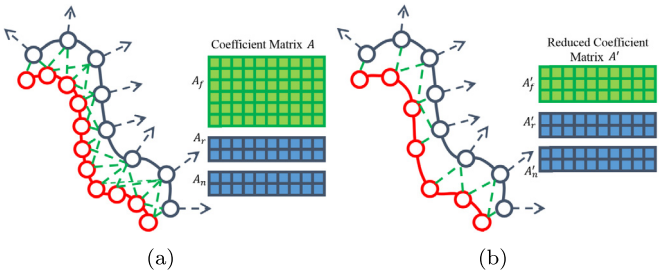


Fig. 3. The points (blue circle) are sampled from input points (red circle), by fitting to their original neighbors (green dashed line) and forcing to be consistent with normals (blue dashed line). Through reducing the number of original neighbors (b), the rows of submatrix A_f can be decreased, resulting in a reduce coefficient matrix A' consequently.

Next, P is optimized with δ fixed:

$$\min_{P}\Big\{\sum_{i\in I}\sum_{q_j\in\mathcal{M}_{p_i}}||p_i-q_j||^2/v_j+\alpha\sum_{i\in I}\sum_{p_{i'}\in\mathcal{N}_{p_i}}||p_i-p_{i'}||^2\bar{\theta}(||p_i-p_{i'}||)/w_i + \beta\sum_{i\in I}\sum_{p_{i'}\in\mathcal{N}_{p_i}}(||\mathbf{n}_i{}^{\mathrm{T}}(p_i-p_{i'})-\delta_{i,i'}||^2)\Big\}. \tag{10}$$

The functions $\bar{\theta}(r)$, v_j and w_i are obtained from the previous iteration, and are treated as constants in the optimization. Hence, this optimization is a sparse quadratic in P, and can be solved by using sparse least-square related techniques. The whole procedure is summarized as Algorithm 1.

The typical way of solving Equation (10) is to treat it as a linear least square problem:

$$\min_{x}||Ax-b||^2, \tag{11}$$

where A is referred as the "coefficient matrix", b is referred as the "objective vector" and x is referred as the "unknowns". A is the combination of three submatrices $A=\{A_f;A_r;A_n\}$, as shown in Fig. 3(a). A_f concerns the fit term to the input points Q, A_r relates to the regularization term and A_n refers to the normal term. The Equation (11) is equivalent to the linear system of equations $(A^{\mathrm{T}}A)x=A^{\mathrm{T}}b$. To solve the equations efficiently, the symmetrical positive definite matrix $A^{\mathrm{T}}A$ can be further factorized using Cholesky decomposition. After forward and back substitution, the unknowns x will be solved successfully.

Algorithm 1 Alternative minimization solver for Eq. (6).

Input: Point set Q with its filtered normals.
Initialize: Downsample Q to get the initial status of S. Compute $\bar{\theta}(r)$, v and w. $\beta\leftarrow 0.1, \mu\leftarrow 1.5, \alpha\leftarrow 0.15$.
1: **repeat**
2: fix P, solve for δ in Eq. (8).
3: fix δ, solve for P in Eq. (10).
4: update $\bar{\theta}(r)$, v and w.
5: $\beta\leftarrow\mu\beta$
6: **until** $\beta>2$
Output: The point set S.

4. Accelerating methods

In Algorithm 1, most of the computation time is spent on solving Equation (11). Point set data usually contains a large size of 3D points, so that it can describe the details of a surface. In the filed of point set processing, it is very common that the size of the point set is larger than hundreds of thousands. Hence, the size of the coefficient matrix A is very large, even in the format of sparse matrix. Moreover, Algorithm 1 takes small steps in the optimization space to approach the true solution cautiously and slowly. It needs to solve Equation (11) about 8 times in total. Although there exist some popular CPU-based sparse least square libraries such as *taucs* (Toledo et al., 2001), *eigen* (Guennebaud et al., 2010) and *MKL PARDISO* (MKL PARDISO Intel, 2008), it takes several hours to process a $10K$ point set with Algorithm 1 using these CPU-based libraries.

We choose a GPU-based least square solver *CGLS* (Saunders et al., 2015), which implements the Conjugate Gradient algorithm for least square with *CUDA* and makes full use of the *cuSPARSE* (The NVIDIA CUDA Sparse Matrix library (cuSPARSE), 2010) and *cuBLAS* (The NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS) library, 2007) libraries. The processing time

is therefore reduced to several minutes, but it is still too long for real applications. Since the computation complexity of the Conjugate Gradient algorithm is superlinear with the size of the matrix A, reducing the size of A could further reduce the processing time.

Our intuition is to find a compact-yet-meaningful representation for the input points Q, so that the size of the submatrix A_f can be greatly reduced. There exist some methods which aim at downsampling the input points. KLOP (Liao et al., 2013) utilizes a kernel density estimate, and CLOP (Preiner et al., 2014) applies a hierarchical Gaussian Mixture Model. As both KLOP and CLOP have closed form solutions in each iteration, their downsampling techniques are not suitable for our case. In this section, we propose a optimization-based local half-sampling method which can reduce the size of A_f while give a close approximation to the original optimization. Besides, we propose a interleaved regularization to further reduce the processing time. As a result, we can accelerate the original method in an order of magnitude.

4.1. Optimization-based local half-sampling

The Equation (10) can be treated as the combination of three terms: the fit term $E_f(p_i)$, the regularization term $E_r(p_i)$ and the normal term $E_n(p_i)$. For each point p_i, our goal is to find a reasonable approximation $\hat{E}_f(p_i)$ of $E_f(p_i)$, using a smaller original neighbors set, denoted by $\hat{\mathcal{M}}_{p_i}$. In other words, we don't downsample the input points Q *as a whole*, but downsample each point's original neighbors set $\mathcal{M}_{p_i}$ *locally*. Mathematically, it can be formulated as the following minimization problem:

$$\min_{\hat{\mathcal{M}}_{p_i}} ||E_f(p_i) - \hat{E}_f(p_i)||^2 = \min_{\hat{\mathcal{M}}_{p_i}} \left\{ \sum_{q_j \in \mathcal{M}_{p_i}} ||p_i - q_j||^2 - \sum_{\hat{q}_j \in \hat{\mathcal{M}}_{p_i}} ||p_i - \hat{q}_j||^2 \right\}, \tag{12}$$

with additional constraints that $|\hat{\mathcal{M}}_{p_i}| \ll |\mathcal{M}_{p_i}|$. Note that the density weights v_j are dropped for brevity, since they are kept as constants in Equation (10).

A feasible way of optimizing $\hat{\mathcal{M}}_{p_i}$ is to seek a spatial partition of $\mathcal{M}_{p_i}$, then the points in the same subset are merged to one point by averaging. Octree (Zhou et al., 2011; Zeng et al., 2013) is a popular method of partitioning point cloud spatially, but it is not designed for optimizing Equation (12). In fact, the partition optimization problem is a combinatorial optimization problem. To reduce the searching space and make the original problem easier to tackle, we add a restrict to the partition that each subset of the partition has just two elements. So we have $\hat{\mathcal{M}}_{p_i} = \{\hat{q}_j | \hat{q}_j = \frac{q_x+q_y}{2} \wedge (q_x, q_y) \in \mathcal{K}_{p_i}\}$, where $\mathcal{K}_{p_i}$ represents the "two-elements partition" of $\mathcal{M}_{p_i}$. Taking it into Equation (12), we arrive at:

$$\min_{\mathcal{K}_{p_i}} \left\{ \sum_{q_j \in \mathcal{M}_{p_i}} ||p_i - q_j||^2 - \sum_{(q_x, q_y) \in \mathcal{K}_{p_i}} 2||p_i - \frac{q_x + q_y}{2}||^2 \right\}. \tag{13}$$

Now our goal becomes finding a two-elements partition that solves Equation (13). After expansion and elimination, the above equation is equivalent to:

$$\min_{\mathcal{K}_{p_i}} \left\{ \sum_{(q_x, q_y) \in \mathcal{K}_{p_i}} ||q_x - q_y||^2 \right\}. \tag{14}$$

The detailed derivation procedure is presented in Appendix A. The meaning of Equation (14) is that, the "best" two-elements partition should satisfy that the sum of the distance of two points within a subset is minimum. The Equation (14) is still a combinatorial optimization problem and hard to solve in polynomial time. We therefore propose a greedy algorithm that is summarized as Algorithm 2 to approximately solve Equation (14). Due to its natural parallel characteristic, Algorithm 2 is easy to be implemented in GPU.

Algorithm 2 A greedy algorithm for solving Eq. (14).

Input: Original neighbors set $\mathcal{M}_{p_i}$.
1: Compute distances between all points in $\mathcal{M}_{p_i}$ and store them as $D = \{(l, q_x, q_y)\}$.
2: Sort the set D according to the distance value l in ascending order.
3: Set indices set empty $E = \{\}$.
4: Set partition empty $\mathcal{K}_{p_i} = \{\}$.
5: **for** each (l, q_x, q_y) in D **do**
6: **if** $x \notin E$ and $y \notin E$ **then**
7: Add (q_x, q_y) to $\mathcal{K}_{p_i}$.
8: Add x and y to E.
9: **end if**
10: **end for**
Output: The partition $\mathcal{K}_{p_i}$.

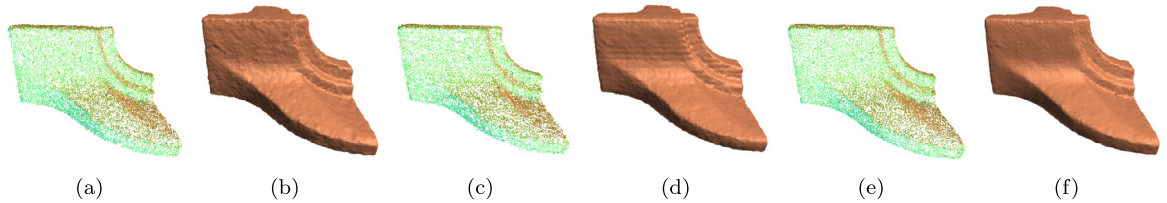


Fig. 4. (a) The number of original neighbors is reduced to 50% by decreasing the search radius by 25%. (c) The original neighbors are half sampled randomly. (e) The original neighbors are half sampled using Algorithm 2. (b), (d) and (f) are the surface reconstruction results of (a), (c) and (e) using Poisson Reconstruction.

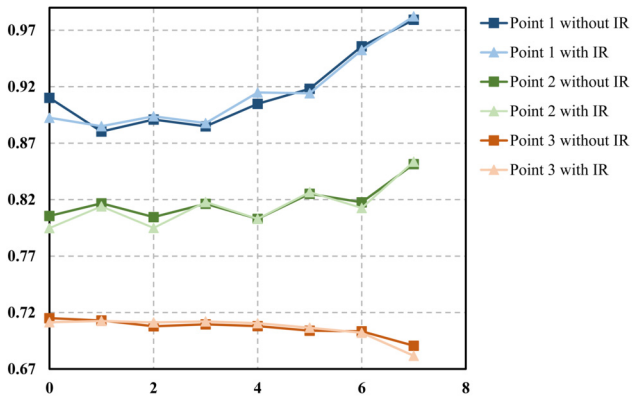


Fig. 5. The X-axis represents the iteration number in Algorithm 1, and the Y-axis represents the local density of a point. IR is the abbreviation of "Interleaved Regularization".

In Fig. 4, we verify the effectiveness of our proposed optimization-based local half-sampling technique by comparison. In Fig. 4(a), the search radius of computing original neighbors is decreased by 25%, so that the original neighbors are cut off by about 50%. In Fig. 4(c), we don't change the search radius but sample 50% of original neighbors randomly. In Fig. 4(e), the search radius is also unchanged, and the original neighbors is half sampled using Algorithm 2. Then the L_0 point set resampling is performed individually for the three types of half reduced original neighbors set. From the reconstructed mesh by Poisson Reconstruction (Kazhdan and Hoppe, 2013), the result of decreasing search radius (Fig. 4(b)) still suffers from noises, and the result of random sampling (Fig. 4(d)) has some stripes. Our optimization-based local half-sampling technique can maintain most structures and details of the original surface. In our experiments, we run two pass of optimization-based local half-sampling, so that the number of a point's original neighbors is quartered. In this setting, we can achieve about $5\times$ faster, including the speed up in solving least square and preparing sparse matrix.

4.2. Interleaved regularization

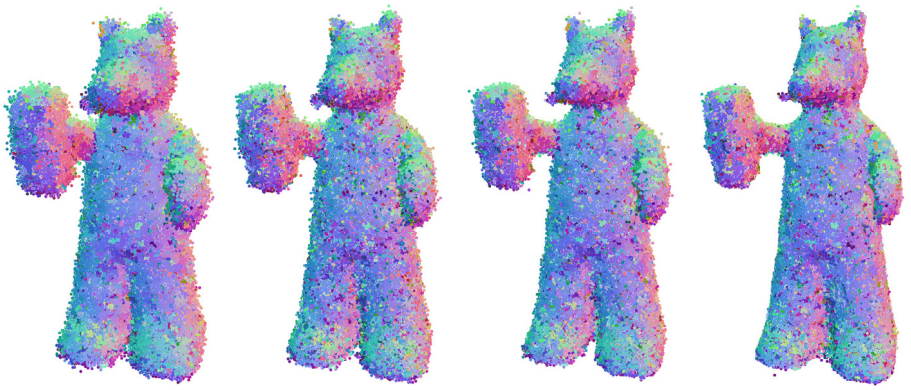
In Algorithm 1, the regularization term is optimized in every iteration, and the weight of the regularization term α is also fixed. In our experiments, we have observed that the change of the local density of a point (computed by Equation (5)) is relatively slow between every two iterations, as shown in Fig. 5. This observation suggests that, we can drop the regularization term in Equation (10) as a rough approximation, and in the next iteration we put back the regularization term. Hence, we can further reduce the computation time by optimizing the regularization term every second iteration. We perform L_0 resampling to the "fandisk" point set with and without interleaved regularization. Then we randomly pick three points from the resulting points, and compute their local density in different iterations. The changes of local density against iterations are shown in Fig. 5, from which we observe that interleaved regularization is a good approximation of full regularization.

5. Results and comparisons

In this section, we evaluate the consolidation quality and the processing efficiency of our proposed L_0 point set resampling (L0R) and accelerated L_0 point set resampling (AL0R). Both L0R and AL0R have only one parameter that need to be specified by users, the normal weight λ. We find that $\lambda = 0.9$ works well for all the testing point set used in this paper, so we keep it fixed. The fast grid-based searching technique is utilized to find original neighbors and self neighbors under a given radius r_p. The setting of r_p is described in Section 3.2. The bilateral normal filtering, least square solver *CGLS* (Saunders et al., 2015), and optimization-based local half-sampling are implemented in GPU. Except for the above three components,

Table 1
Model statistics and individual timings in seconds.

Model	Fandisk		Armadillo		Lord Quas		Anchor		FIFA		Box		Clock	
$\|Q\|$	110K		100K		160K		120K		130K		200K		300K	
$\|P\|$	60K		80K		100K		83K		90K		200K		220K	
Timing	L0R	AL0R	L0R	AL0R	L0R	AL0R	L0R	AL0R	L0R	AL0R	L0R	AL0R	L0R	AL0R
Init	2.1	2.1	3.8	3.8	4.2	4.2	2.1	2.1	2.7	2.7	4.8	4.8	5.9	5.9
BF	3.5	3.5	4.7	4.7	5.6	5.6	3.2	3.2	4.8	4.8	6.0	6.0	6.7	6.7
OLHS		2.1		4.5		8.5		1.9		4.8		12.5		14.4
L_0	350.6	42.5	956.1	99.6	2112.3	149.7	537.5	55.3	910.7	88.1	4213.4	513.7	5350.0	584.4
Total	356.2	50.2	964.6	112.6	2122.1	168.0	542.8	62.5	918.2	100.4	4224.4	537.0	5362.6	611.4
Speedup	7.1		8.6		12.6		8.7		9.1		7.8		8.8	



(a) Noisy input (b) $\lambda = 0.05$ (c) $\lambda = 0.3$ (d) $\lambda = 0.9$

Fig. 6. The impact of normal weight λ. The noisy "Lord Quas" point set ($160K$) is resampled by AL0R with different λ. With the increase of λ, more noises and outliers are removed.

all other components are implemented in CPU using C/C++. All results were produced on a PC with an Intel i7-4770 3.4 GHz CPU and a NVIDIA GeForce GTX 780 GPU.

Table 1 summarizes statistics and processing time (seconds) for the seven testing point set, which contains both synthetic data (Fandisk, Armadillo, Lord Quas and Anchor) and real-world data (FIFA, Box and Clock). Each synthetic point set is corrupted by Gaussian noise with a standard deviation that is about 1%-2% of the diagonal length of the bounding box. The real-world data is scanned by a laser scanner. $|Q|$ denotes the number of points in the input data Q, while $|P|$ denotes the number of points in the output data P. The total processing time of a point set includes initialization (Init), bilateral normal filtering (BF), optimization-based local half-sampling (OLHS, optional) and L_0 resampling (L_0). The procedure of initialization usually contains neighbors searching and initial density computing. Both L0R and AL0R use the GPU-based least square solver (Saunders et al., 2015). The numbers of speed up in the last row demonstrate that AL0R achieves an overall 10 times faster than L0R.

The compared point set resampling methods include LOP (Lipman et al., 2007), WLOP (Huang et al., 2009), CLOP (Preiner et al., 2014), PPU (Wang et al., 2019), EAR (Huang et al., 2013b) and L0D (Sun et al., 2015). The implementations of LOP, WLOP and EAR are all from the software developed by Huang et al. (2009, 2013b, 2013a), which integrates the above three methods in a single platform. The implementation of CLOP and PPU are obtained from their original authors. Regrading L0D, we implement it by ourselves. For all the compared methods, we strictly follow the guidelines provided by the distributed codes and use the suggested parameters in their paper. For each testing point set, the number of resampled points $|P|$ are all the same in all methods. Note that the EAR scheme firstly downsamples a point set and then upsamples it. Hence, when using EAR to resample the point set of size M, we need to downsample it to $2M/3$ firstly and then add $M/3$ projected points.

In Fig. 1, our method shows superior performance in terms of robustness to noise and effective recovery of edge features. Moreover, our method can further enhance the performance of the current most popular surface reconstruction method, Poisson Reconstruction (Kazhdan and Hoppe, 2013). As LOP, WLOP and PPU don't incorporate normal information, we use the bilateral filtered normals by our method to render their resampled points and help their surface reconstruction. In Fig. 6, we show the impact of normal weight λ. In Figs. 7, 8, 9 and 10, we show more comparisons among PPU, EAR, L0D, L0R and AL0R. The surface reconstruction results continue to be utilized to help us assess the quality of point set resampling. It is evident that, compared with PPU, EAR and L0D, both L0R and AL0R have an advantage in piecewise smooth, uniform distribution and feature preserving reconstruction in almost all scenarios. Although AL0R reduces the number of original neighbors and skips regularization optimization every second iteration, it is able to produce a comparable or even better

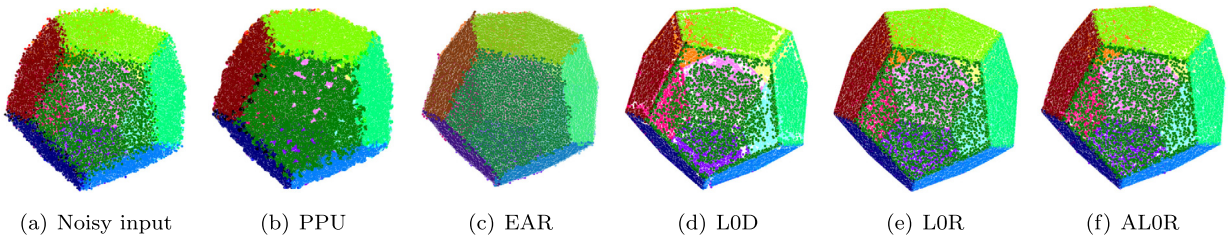


Fig. 7. Comparison in the "Dodecahedron" point set ($43K$). The noisy input (a) is resampled by PPU (Wang et al., 2019) (b), EAR (Huang et al., 2013b) (c), L0D (Sun et al., 2015) (d), L0R (e) and AL0R (f) respectively. For better visual effect, all point sets (a-e) are colored by the bilateral filtered normals.

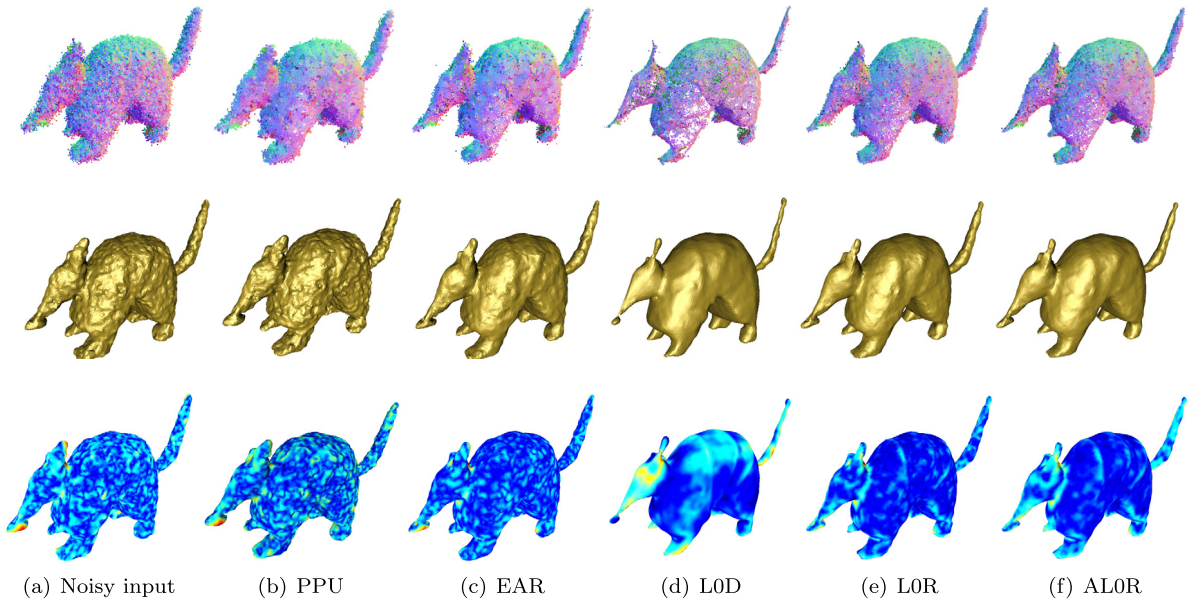


Fig. 8. Quality comparison in the "Armadillo" point set ($100K$). The noisy input (a) is resampled by PPU (Wang et al., 2019) (b), EAR (Huang et al., 2013b) (c), L0D (Sun et al., 2015) (d), L0R (e) and AL0R (f) respectively, and the resampled points are shown in the first row. Poisson Reconstruction (depth = 7) is performed for the five set of resampled points individually, and the reconstruction results are presented in the second row. Errors between each Poisson Reconstruction result and the ground truth are visualized in the third row, where the red color indicates large error and the blue color indicates small error.

reconstruction than L0R. For example, in Fig. 8, the surface reconstruction results by AL0R are more smooth than L0R, which indicates that the optimized-based local half-sampling has a certain ability of removing noises.

Comparison with PPU (Wang et al., 2019). Although PPU is the most recent neural network based resampling method, we found that PPU cannot produce satisfactory result if the input is particularly noisy and messy, as shown in Fig. 1, 7, 8 and 9. The network used in PPU maps the raw 3D coordinates to a feature space, expand and interpolate the features in a multi-step manner. Neither noises removing or feature preserving is considered in the feature extraction, expansion, interpolation and loss computation explicitly.

Comparison with EAR (Huang et al., 2013b). The main spirit of EAR is bilateral filtering, including a bilateral normal filter which separates normals around edges and a bilateral projection operator which upsamples points progressively. L_0 minimization has stronger ability in feature-preserving than bilateral filtering, which has been carefully discussed in the literature (Xu et al., 2011; Cheng et al., 2014; He and Schaefer, 2013). In Fig. 11, we present the comparisons between EAR and AL0R in handling real-world data which contain holes and have uniform point distribution. Readers can zoom in to see more sharpness of the ALOR's reconstructed results than EAR's.

Comparison with L0D (Sun et al., 2015). Our method is very closed to L0D. In fact, L0D is the combination of L_0 minimization and EAR upsampling (Huang et al., 2013b). Since EAR upsampling can increase sharp features greatly, it is hard to judge L_0 minimization or EAR upsampling take effects. Hence, our implementation of L0D doesn't include EAR upsampling. As shown in Fig. 1 and 7, L0D can reconstruct sharp features, but cannot get uniform distribution. Our method is also based on L_0 minimization, but incorporates density weights and terms. We argue that L_0 minimization can get uniformly distributed results independently, without EAR upsampling. Besides, our method is far more fast than L0D, benefited from our accelerating methods.

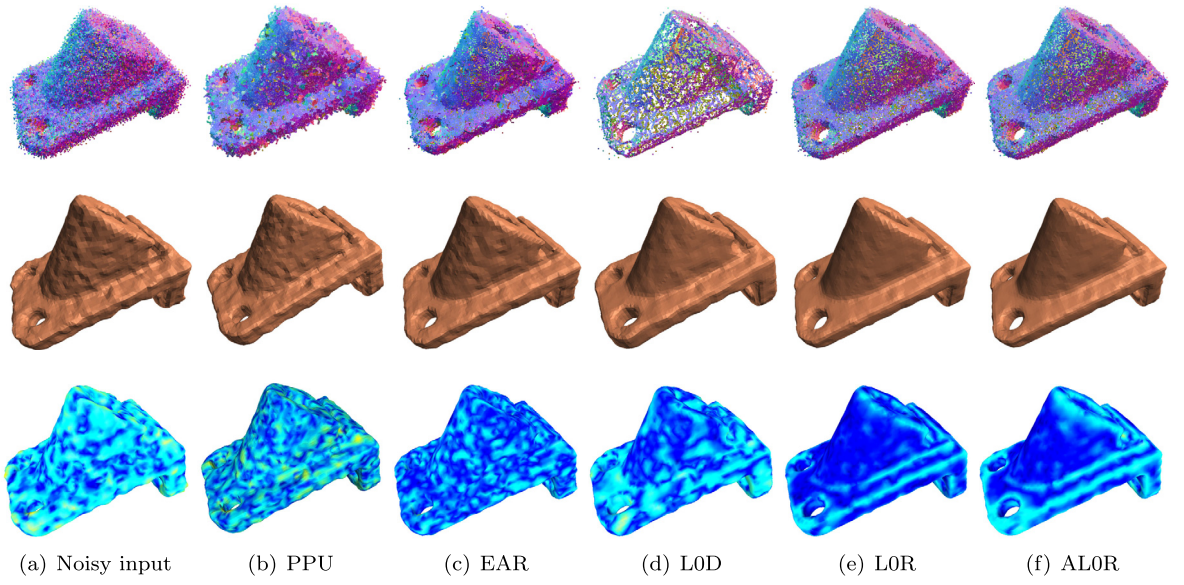


Fig. 9. Quality comparison in the "Anchor" point set (120K). The noisy input (a) is resampled by PPU (Wang et al., 2019) (b), EAR (Huang et al., 2013b) (c), L0D (Sun et al., 2015) (d), L0R (e) and AL0R (f) respectively, and the resampled points are shown in the first row. Poisson Reconstruction (depth = 6) is performed for the five set of resampled points individually, and the reconstruction results are presented in the second row. Errors between each Poisson Reconstruction result and the ground truth are visualized in the third row, where the red color indicates large error and the blue color indicates small error.

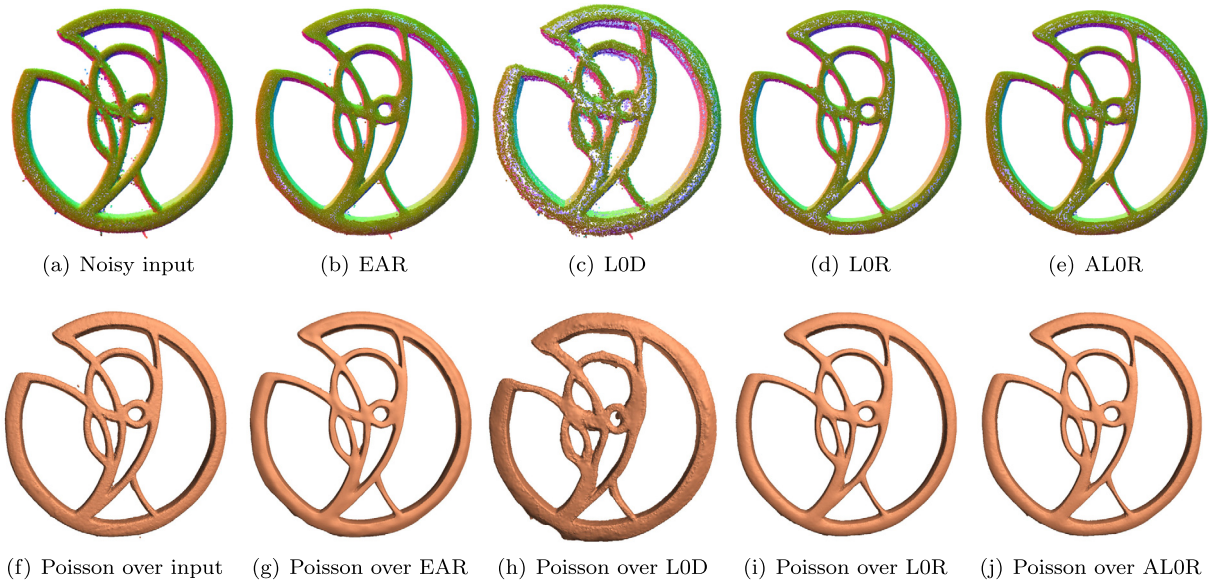


Fig. 10. Quality comparison in the "FIFA" point set (130K). The noisy input (a) is resampled by EAR (Huang et al., 2013b) (b), L0D (Sun et al., 2015) (c), L0R (d) and AL0R (e) respectively. Poisson Reconstruction (depth = 9) is performed on the four sets of resampled points individually, and the reconstruction results are presented in (e-h).

6. Limitations

Although our method has shown strengths in handling noises and preserving features, failure cases can still occur when dealing with certain type of thin structures. We show an example in Fig. 12, in which some holes appear in the reconstructed mesh model. The points on the two sides of the vertical plate have exactly opposite normal orientation. Due to the close distances between these points, our L_0 resampling would mix them together, subsequently causing errors in the surface reconstruction. Besides, like other normal-based resampling methods (Huang et al., 2013b; Avron et al., 2010; Sun et al., 2015), our method needs the initial normal computed by PCA to have right orientation. Otherwise, our L_0 resampling would also fail due to the wrong normal orientation. Finally, our method does not address the missing data problem.

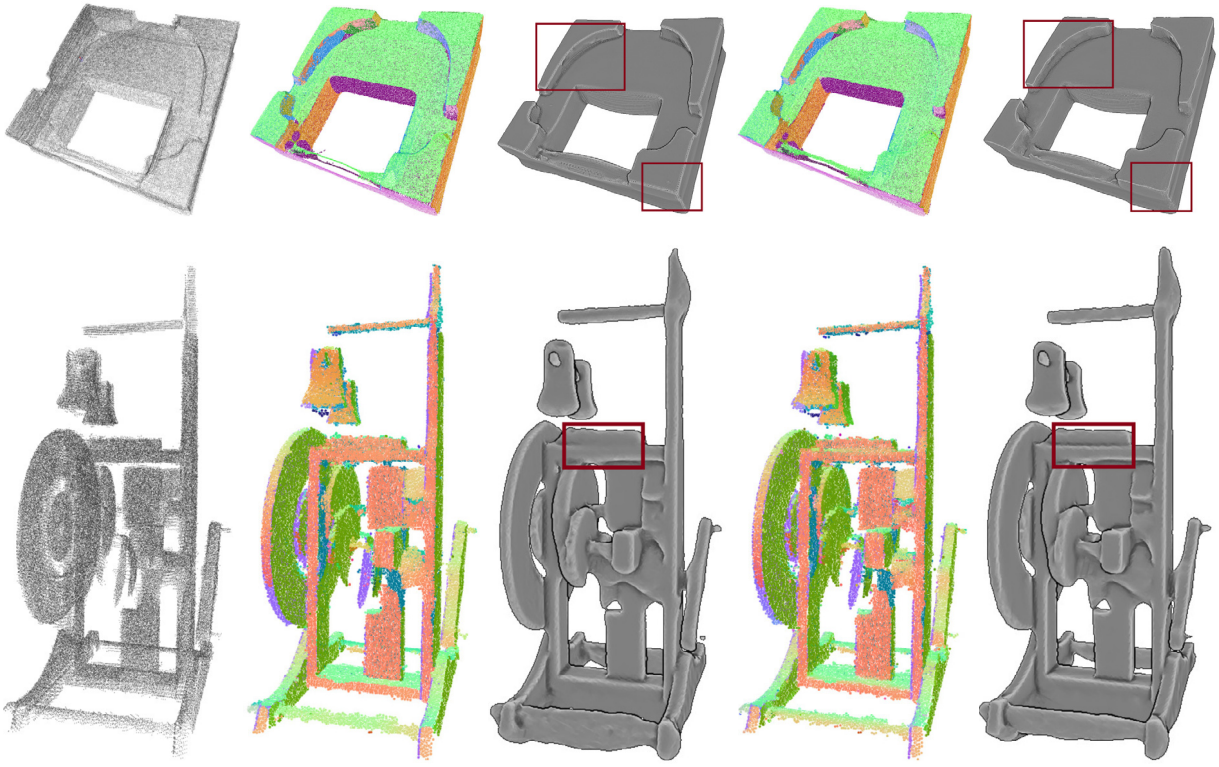


Fig. 11. Comparisons between EAR (Huang et al., 2013b) and AL0R in sharp features reconstruction. The input scan "Box" ($200K$) and "Clock" ($300K$) are resampled by EAR (Huang et al., 2013b) (b) and AL0R (d) respectively. Poisson Reconstruction (depth = 8) is performed on the resampled points individually, and the reconstruction results are presented in (c) and (e). The edges on each reconstructed mesh model are enhanced using radiance scaling, and readers can zoom in to evaluate the sharpness of the reconstructed models.

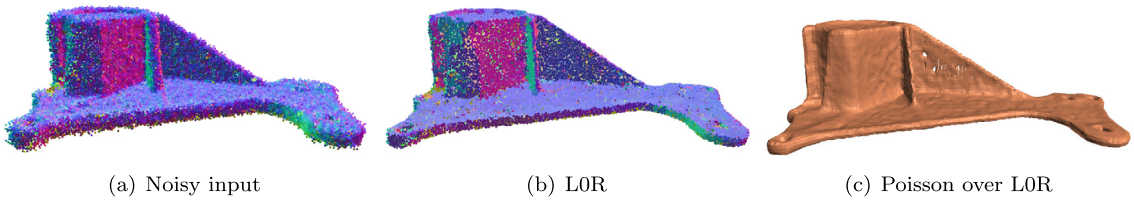


Fig. 12. A failure case. Some holes appear in the vertical plate of the reconstructed mesh model (c).

7. Conclusions

We have proposed a novel point set resampling method in an L_0 minimization framework. Our framework can produce a set of clean, uniformly distributed, geometry-maintaining and feature-preserving oriented points from an unorganized and noisy point set. Compared with current point set resampling methods, our method specializes in anti-noise and persevering features based on the L_0 norm constraints. To further speed up our method, we also provide two accelerating algorithms. Optimization-based local half-sampling can reduce the number of a point's neighbors, while offers a close approximation to the original optimization function. Interleaved regularization further increases the efficiency by performing regularization in every two iterations.

Although our accelerated L_0 point set resampling achieve about an order of magnitude faster, it is still far from interactive applications (order of milliseconds). Most efforts of our increasing efficiency are spent on how to reduce the data size and how to find a approximate optimization, with little attention on how to improve the alternative minimization solver (Xu et al., 2011) itself. In the future, we plan to develop a more efficient algorithm to replace the alternative minimization solver (Xu et al., 2011), and find a closed form solution for the quadratic problem in Equation (10), making L_0 point set resampling in interactive rate become reality. Besides, we also plan to apply L_0 minimization in the loss function of neural network based points resampling (Yu et al. 2018a, 2018b; Wang et al., 2019) and Centroidal Voronoi Tessellation based points resampling (Chen et al., 2018).

Declaration of competing interest

None.

Acknowledgements

Many thanks to the editor and the anonymous reviewers for their valuable comments. This work was partially supported by the National Natural Science Foundation of China (No. 61802322, No. 61402387 and No. 61602139), the Guiding Project of Fujian Province, China (No. 2018H0037), the Fundamental Research Funds for the Central Universities, China (No. 20720190003), and the Research Funds administered by the Digital Fujian, at the Big Data Institute for Urban Public Safety.

Appendix A. Derivation from Equation (13) to (14)

$$\begin{aligned}
&\sum_{q_j\in\mathcal{M}_{p_i}}||p_i-q_j||^2-\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}2||p_i-\frac{q_x+q_y}{2}||^2\\
&=\left\{|\mathcal{M}_{p_i}|\cdot||p_i||^2+\sum_{q_j\in\mathcal{M}_{p_i}}||q_j||^2-2p_i\cdot\sum_{q_j\in\mathcal{M}_{p_i}}q_j)\right\}\\
&-\left\{2|\mathcal{K}_{p_i}|\cdot||p_i||^2+\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}\frac{||q_x+q_y||^2}{2}-2p_i\cdot\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}(q_x+q_y)\right\}
\end{aligned}$$

Noting that $|\mathcal{M}_{p_i}|=2|\mathcal{K}_{p_i}|$ and $\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}(q_x+q_y)=\sum_{q_j\in\mathcal{M}_{p_i}}q_j$, we get

$$\begin{aligned}
&\sum_{q_j\in\mathcal{M}_{p_i}}||p_i-q_j||^2-\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}2||p_i-\frac{q_x+q_y}{2}||^2\\
&=\sum_{q_j\in\mathcal{M}_{p_i}}||q_j||^2-\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}\frac{||q_x+q_y||^2}{2}\\
&=\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}(||q_x||^2+||q_y||^2)-\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}\frac{||q_x+q_y||^2}{2}\\
&=\sum_{(q_x,q_y)\in\mathcal{K}_{p_i}}\frac{||q_x-q_y||^2}{2}
\end{aligned}$$

Drop the factor of $\frac{1}{2}$, we finally arrive at Equation (14).

References

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T., 2003. Computing and rendering point set surfaces. IEEE Trans. Vis. Comput. Graph. 9 (1), 3–15.
Amenta, N., Choi, S., Dey, T.K., Leekha, N., 2002. A simple algorithm for homeomorphic surface reconstruction. Int. J. Comput. Geom. Appl. 12 (1–2), 125–141.
Avron, H., Sharf, A., Greif, C., Cohen-Or, D., 2010. L_1-sparse reconstruction of sharp point set surfaces. ACM Trans. Graph. 29 (5), 135.
Berger, M., Levine, J.A., Nonato, L.G., Taubin, G., Silva, C.T., 2013. A benchmark for surface reconstruction. ACM Trans. Graph. 32 (2), 20.
Brandt, C., Seidel, H., Hildebrandt, K., 2015. Optimal spline approximation via L_0 minimization. Comput. Graph. Forum 34 (2), 617–626.
Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R., 2001. Reconstruction and representation of 3D objects with radial basis functions. In: Proc. of SIGGRAPH.
Chen, Z., Zhang, T., Cao, J., Zhang, Y.J., Wang, C., 2018. Point cloud resampling using centroidal Voronoi tessellation methods. Comput. Aided Des. 102, 12–21.
Cheng, X., Zeng, M., Liu, X., 2014. Feature-preserving filtering with L_0 gradient minimization. Comput. Graph. 38, 150–157.
Dey, T.K., Giesen, J., 2001. Detecting undersampling in surface reconstruction. In: Proc. of Symposium on Computational Geometry.
Dey, T.K., Wang, L., 2013. Voronoi-based feature curves extraction for sampled singular surfaces. Comput. Graph. 37 (6), 659–668.
Guennebaud, G., Jacob, B., et al., 2010. Eigen v3. http://eigen.tuxfamily.org.
He, L., Schaefer, S., 2013. Mesh denoising via L_0 minimization. ACM Trans. Graph. 32 (4), 64.
Hoppe, H., DeRose, T., Duchamp, T., McDonald, J.A., Stuetzle, W., 1992. Surface reconstruction from unorganized points. In: Proc. of SIGGRAPH.
Huang, H., Li, D., Zhang, H., Ascher, U.M., Cohen-Or, D., 2009. Consolidation of unorganized point clouds for surface reconstruction. ACM Trans. Graph. 28 (5), 176.
Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., Chen, B., 2013a. L_1-medial skeleton of point cloud. ACM Trans. Graph. 32 (4), 65.
Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U.M., Zhang, H.R., 2013b. Edge-aware point set resampling. ACM Trans. Graph. 32 (1), 9.
Kazhdan, M.M., Hoppe, H., 2013. Screened Poisson surface reconstruction. ACM Trans. Graph. 32 (3), 29.
Kazhdan, M.M., Bolitho, M., Hoppe, H., 2006. Poisson surface reconstruction. In: Proc. of Eurographics Symposium on Geometry Processing.
Kolluri, R.K., Shewchuk, J.R., O'Brien, J.F., 2004. Spectral surface reconstruction from noisy point clouds. In: Proc. of Symposium on Computational Geometry.
Liao, B., Xiao, C., Jin, L., Fu, H., 2013. Efficient feature-preserving local projection operator for geometry reconstruction. Comput. Aided Des. 45 (5), 861–874.

Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H., 2007. Parameterization-free projection for geometry reconstruction. ACM Trans. Graph. 26 (3), 22.
MKL PARDISO Intel, 2008. https://software.intel.com/en-us/node/470282.
Öztireli, A.C., Guennebaud, G., Gross, M.H., 2009. Feature preserving point set surfaces based on non-linear kernel regression. Comput. Graph. Forum 28 (2), 493–501.
Preiner, R., Mattausch, O., Arikan, M., Pajarola, R., Wimmer, M., 2014. Continuous projection for fast L_1 reconstruction. ACM Trans. Graph. 33 (4), 47.
Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017a. PointNet: deep learning on point sets for 3D classification and segmentation. In: Proc. of CVPR.
Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017b. PointNet++: deep hierarchical feature learning on point sets in a metric space. In: Proc. of NIPS.
Saunders, M., Hansen, P.C., Bleichrodt, F., Fougner, C., 2015. CGLS: CG method for Ax = b and least squares. http://web.stanford.edu/group/SOL/software/cgls/.
Sun, Y., Schaefer, S., Wang, W., 2015. Denoising point sets via L_0 minimization. Comput. Aided Geom. Des. 35, 2–15.
The NVIDIA CUDA Basic Linear Algebra Subroutines (cuBLAS) library, 2007. https://developer.nvidia.com/cuBLAS.
The NVIDIA CUDA Sparse Matrix library (cuSPARSE), 2010. https://developer.nvidia.com/cuSPARSE.
Toledo, S., Chen, D., Rotkin, V., 2001. TAUCS: a library of sparse linear solvers. http://www.tau.ac.il/~stoledo/taucs/.
Wang, Q., Tao, Y., Lin, H., 2015. Edge-aware volume smoothing using L_0 gradient minimization. Comput. Graph. Forum 34 (3), 131–140.
Wang, Y., Wu, S., Huang, H., Cohen-Or, D., Sorkine-Hornung, O., 2019. Patch-based progressive 3D point set upsampling. In: Proc. of CVPR.
Xu, L., Lu, C., Xu, Y., Jia, J., 2011. Image smoothing via L_0 gradient minimization. ACM Trans. Graph. 30 (6), 174.
Yu, L., Li, X., Fu, C., Cohen-Or, D., Heng, P., 2018a. PU-Net: point cloud upsampling network. In: Proc. of CVPR.
Yu, L., Li, X., Fu, C., Cohen-Or, D., Heng, P., 2018b. EC-Net: an edge-aware point set consolidation network. In: Proc. of ECCV.
Zeng, M., Zhao, F., Zheng, J., Liu, X., 2013. Octree-based fusion for realtime 3D reconstruction. Graph. Models 75 (3), 126–136.
Zhang, W., Jiang, H., Yang, Z., Yamakawa, S., Shimada, K., Kara, L.B., 2018. Data-driven upsampling of point clouds. arXiv preprint arXiv:1807.02740.
Zhou, K., Gong, M., Huang, X., Guo, B., 2011. Data-parallel octrees for surface reconstruction. IEEE Trans. Vis. Comput. Graph. 17 (5), 669–681.