Noname manuscript No.
(will be inserted by the editor)

A PDE Patch-based Spectral Method for Progressive Mesh Compression and Mesh Denoising

Qiqi Shen · Yun Sheng · Congkun Chen · Guixu Zhang · Hassan Ugail

Received: date / Accepted: date

Abstract The development of the patchwise Partial Differential Equation (PDE) framework a few years ago has paved the way for the PDE method to be used in mesh signal processing. In this paper we, for the first time, extend the use of the PDE method to progressive mesh compression and mesh denoising. We, meanwhile, upgrade the existing patchwise PDE method in patch merging, mesh partitioning, and boundary extraction to accommodate mesh signal processing. In our new method an arbitrary mesh model is partitioned into patches, each of which can be represented by a small set of coefficients of its PDE spectral solution. Since low-frequency components contribute more to the reconstructed mesh than high-frequency ones, we can achieve progressive mesh compression and mesh denoising by manipulating the frequency terms of the PDE solution. Experimental results demonstrate the feasibility of our method in both progressive mesh compression and mesh denoising.

Keywords Spectral method · Mesh processing · Patchwise PDE · Progressive mesh compression · Mesh denoising

1 Introduction

Since Bloor and Wilson's pioneering work [3] first applied Partial Differential Equations (PDEs) to generating blend surface decades ago, advantages of the PDE method in computer graphics have been gradually discovered. One of the main advantages comes from the ability that the differential operators of PDEs can generate smooth surfaces. Another major advantage is that a 3D surface can be generated by manipulating a relatively small set of boundary curves. These advantages enable the PDE method to be applied to many research fields, such as surface modeling [4] and computer-aided manufacturing [5,10] in the 1990s, and shape morphing [6], Web visualisation [28], interactive design [37], face parameterisation [33], pharmaceutical modeling [1], and medical image visualisation [7] after the millennium.

In the Bloor and Wilson's PDE (BWPDE) method, a 3D parametric PDE patch is defined as a solution to an elliptic PDE that is analytically resolved by imposing Fourier analysis on its boundary conditions. In this sense, the PDE method possesses some spectral characteristics due to the involvement of Fourier analysis, but these spectral characteristics of the PDE method have never been explored previously, mainly because the PDE method, all the time leveraged as a modeling tool of smooth surfaces, cannot be directly used to approximate irregular and sharp geometric details of a given surface.

In order to resolve this problem, Sheng *et al.* [32] proposed a patchwise PDE (PPDE) method, the main distinction of which from the BWPDE method is the localisation of the coordinate system for each PDE patch. Although the development of the PPDE method has paved the way for spectral processing of irregular geometric meshes, some issues in the PPDE method, such as patch merging, mesh partitioning, and boundary extraction, are yet to be studied to accommodate spectral analysis.

In this paper we, for the first time, explore its spectral nature of the PPDE method in mesh signal process-

Qiqi Shen · Yun Sheng (✉) · Congkun Chen · Guixu Zhang
The School of Computer Science and Software Engineering, East China Normal University, 200062, Shanghai, China.
E-mail: ysheng@cs.ecnu.edu.cn

Hassan Ugail
The Centre for Visual Computing, University of Bradford, Bradford BD7 1DP, UK.

ing, more specifically, in progressive mesh compression and mesh denoising. To accommodate progressive mesh compression and mesh denoising we upgrade the PPDE method as follows. First, instead of manually selecting position fixers for PDE patch merging [32], we introduce a new merging scheme, which blends the reconstructed PDE patches in terms of point cloud before the blended point cloud is globally triangulated. The new merging scheme enables the PDE patches to be seamlessly blended together without user supervision. Second, instead of partitioning the mesh with a simplification algorithm [32] that may give rise to geometry loss, we adopt MeTiS [22] for mesh partitioning. Patches segmented by MeTiS retain both geometry and topology of the original mesh. Third, the boundary extraction method used in [32] was tailored for triangle-shaped PDE patches, while ours in this paper is designed for MeTiS-partitioned patches with arbitrary shapes.

The philosophy behind our method is stated as follows. By imposing Fourier analysis on the PDE boundary conditions the mesh geometry can be handled in the frequency domain. By taking advantage of the spectral property that low-frequency components contribute more to the reconstructed mesh than the high-frequency ones in terms of geometric structure, mesh signal processing can be performed. In mesh progressive compression, we first transmit the low-frequency coefficients of PDE solutions for a coarse mesh, followed by transmitting the high-frequency coefficients and at the receiving end, gradually recovering more geometric details of the mesh. In mesh denoising, a mesh with noise is transformed into the spectral domain by Fourier analysis of the PPDE method, and we can achieve low-pass filtering by retaining the low-frequency components while discarding the high-frequency ones. Iterating the above process, we can obtain a desired denoising result. It is worth noting that our primary goal is not to propose new progressive mesh compression and mesh denoising methods exceeding the existing ones, although our new methods outperform them to some extent.

The rest of the paper is structured as follows. The related work on progressive mesh compression, mesh denoising as well as PDE methods is briefed in Section 2. Section 3 introduces the PPDE method. Section 4 is dedecated to the mesh preprocessing technologies used, including mesh partitioning and boundary extraction. Section 5 describes our progressive mesh compression scheme. Section 6 states our mesh denoising scheme. Section 7 lends experimental evaluations to our method. Section 8 concludes the paper.

2 Related Work

Both progressive mesh compression and mesh denoising have been playing crucial roles in 3D mesh signal processing. In this section, we briefly review both the research fields as well as the mainstream PDE methods.

2.1 Progressive Mesh Compression

Progressive mesh compression has been studied for years, the existing methods of which can be grouped into spatial and spectral methods. The first progressive mesh compression method introduced by Hoppe [18] is a spatial method, where he used edge collapse and vertex split operations to achieve progressive compression. In order to make the compression ratio of [18] closer to those of single-rate methods, Taubin *et al* [36] introduced a progressive algorithm through the forest split operation, while Pajarola and Rossignac [27] improved Hoppe's method by grouping vertex splits into batches. Using the above two methods, a lower compression rate can be obtained for meshes with acceptable quality. In fact, progressive mesh compression is to find a balance between the restored mesh quality and compression rate, namely the rate-distortion trade-off. For this purpose, Lee *et al.* [26] proposed a rate-distortion optimization (RDO) algorithm with adaptive quantization. All the above are of connectivity-driven algorithms that first encode connectivity data and then use them to encode geometry data. Instead, geometry-driven algorithms proceed in an opposite way. Since geometry data take up more storage space than connectivity data, geometry compression is generally more efficient than connectivity compression. Gandoin and Devillers [14] focused their effort on geometry compression by proposing a geometry-driven algorithm based on the KD-tree subdivision, while Peng and Kuo [30] proposed a geometry-driven algorithm using the octree subdivision.

Different from all the above spatial methods, the methods focusing on spectral processing can build a good approximation of the original mesh and achieve a lower compression rate in a global manner. Karni and Gotsman [20] proposed a spectral method, where mesh geometry was projected onto an orthonormal basis of the Laplacian matrix constructed with the mesh topological information, and then the corresponding eigenvalues were treated as frequency components. As the computational complexity of eigenvectors of an $n \times n$ matrix is $O(n^3)$, to reduce the complexity the input mesh was partitioned into a series of submeshes [20], and each submesh was compressed independently. Nevertheless, the eigenvectors were also required to com-

pute at the decoder, and thus the computational complexity of this method was too heavy for real-time applications. To overcome this problem, Karni and Gotsman [21] proposed to use fixed basis functions rather than variable basis functions for spectral compression of 3D meshes. In fact, these fixed basis functions are Fourier basis functions so that the process of encoding and decoding can be efficiently implemented using the Fast Fourier Transform. Valette and Prost [38] proposed a compression scheme based on wavelet decomposition, where the connectivity of a higher level mesh was reconstructed with that of the base mesh and subdivisions, and the mesh geometry was subsequently reconstructed with wavelet coefficients.

2.2 Mesh Denoising

The literature in mesh denoising can also be grouped into spatial and spectral methods. The first spectral method for mesh fairing based on the graph Laplacian was proposed by Taubin [35]. In this method, vertexes of a mesh were treated as a 3D signal and defined over the underlying mesh graph. Since then, more and more attention has been paid to smoothing the mesh within the spectral domain. Desbrun *et al* [11] proposed an implicit fairing method, where a new scale-dependent umbrella operator was used for avoiding large distortions on irregular meshes. Zhang *et al* [40] proposed efficient techniques to address the computational difficulties of Butterworth filtering and implicit fairing for irregular meshes. For the Butterworth filter, they proposed that factorizing the liner system in the complex domain could accelerate the computation speed. As for implicit fairing, they used successive overrelaxation to improve the processing speed. Kim *et al* combined the explicit [35] and implicit [11,40] forms together to construct a more flexible second order filter, named GeoFilter [24], in which frequencies were computed automatically according to user selected features so as to achieve a desired result. Pauly and Gross [29] proposed a spectral filtering framework for point-sampled geometric models with noise, where a noised model was split into a number of surface patches with regularized samples, before Fourier analysis was applied to removing the noise.

All the above methods are implemented in the spectral domain. Nonetheless, both noise and geometric details in the mesh model correspond to high-frequency components in the spectral domain. Some geometric details are also inevitably removed during denoising. In order to remove noise while retain more geometric details of the mesh, some spatial methods have been developed. For example, Fleishman *et al* [13] and Jones *et al* [19] developed bilateral filters based on vertex positions of the mesh, while Sun *et al* [34] and Zheng *et al* [44] applied bilateral filters to face normals, followed by updating the vertex positions according to the filtered normals. The effectiveness of a bilateral filter usually relies on the range kernel that influences the weights, and the range weights are determined by the intensity difference of the input signal. Nevertheless, the input signal as the guidance sometimes cannot achieve desirable results, and this leads to the development of the joint bilateral filter. Zhang *et al* proposed a guided mesh normal filter [43] for joint bilateral filtering of geometry signal, where a properly constructed normal field was used as the guidance, and the joint bilateral filter was applied to the face normals followed by updating the vertex positions. In order to effectively denoise 3D models with variant levels of noise, Lu *et al* proposed two robust mesh denoising approachs [15] [16]. In [15] an initial estimation is introduced to largely reduce the noise level, followed by a string of operations to preserve features during denoising. In [16] a three-step method is proposed, consisting of vertex pre-filtering of input noise, L_1-median filtering of face normals, vertex position updating according to the filtered normals. In addition to the above-mentioned methods, sparsity optimisation is also popular in mesh denoising. For example, He and Schaefer [17] adopted L_0 minimization to remove noise from meshes, and demonstrated its effectiveness in preserving sharp features.

2.3 The PDE Methods

The mainstream PDE methods in geometric modeling are resolved either numerically or analytically. The analytic solution is suitable for PDEs with closed boundary conditions. Otherwise, a numerical solution has to be sought, usually computationally more expensive. The pioneering PDE method proposed by Bloor and Wilson [3] was analytically resolved by imposing Fourier-analysis on the boundary conditions of a biharmonic-like fourth order PDE. Zhang and You also proposed to analytically resolve a fourth order PDE based on the Pseudo-Levy Series [42,39]. In order to reach a trade-off between the surface smoothness and computational complexity, a fourth-order PDE is generally chosen. However, it can only ensure a C^1 continuity between PDE patches. If a higher continuity is required a higher order PDE has to be employed [41,25].

All the methods surveyed above are based on an analytic solution to resolve PDEs. Since not every PDE has an analytic solution, seeking a numerical solution can improve the generality of a PDE method. For example, Du and Qin [12] proposed to use a finite difference method to resolve PDEs.

Although the above includes various PDE methods, the BWPDE method has had a widest spectrum of applications due to its generality and computational efficiency.

3 The PPDE Method

The PDE method adopted in this paper is a patchwise generalization of the BWPDE method. In the BWPDE method, a 3D parametric PDE patch $S(u,v)$ is defined as a solution to a biharmonic-like fourth order elliptic PDE:

$$\left(\frac{\partial^2}{\partial u^2}+a^2\frac{\partial^2}{\partial v^2}\right)^2 S(u,v)=0 \tag{1}$$

where $0 \le u \le 1$, $0 \le v \le 2\pi$, and a is called the smoothing parameter, governing the relative rate of smoothing between the u and v directions. The fourth-order PDE is adopted because a lower-order PDE has no freedom to specify the reconstructed surface smoothness, whereas the calculation of a higher-order PDE is more time-consuming.

The smooth nature of the PDE is reflected by the partial differential operators in (1), in which the value of the function at any point on the surface is, in a certain sense, a weighted average of the surrounding values. Thus, a surface is obtained as a smooth transition between the boundary conditions. However, this smooth nature makes the BWPDE method impossible to approximate irregular and sharp geometric details of the original surface.

In order to approximate surfaces with irregular and sharp details, Sheng *et al* [32] proposed a PPDE method. In the BWPDE method, boundary curves are extracted along the surface in a consistent order, so that each PDE patch is adjacent to two other patches. In such a patch configuration, each PDE patch shares a globalized uv parametric coordinate system with the others. Due to the smooth nature of the PDE method, such a patch configuration smoothes out some details between the boundary curves extracted from the original surface, which is, however, undesired in geometry approximation. On the contrary, in the PPDE method, a surface is divided into a number of patches. According to the size and orientation of each patch, a local uv coordinate system, independent of the other patches, is assigned to each patch. Eventually, all the individual patches are blended to approximate the original surface. Such a patch configuration enables approximation of irregular and sharp geometric details of the original surface. Our spectral method is built upon the PPDE method, and PDE patch generation and patch merging are discussed in detail as follows.

3.1 PDE Patch Generation

An original surface can be divided into a number of patches, and each patch $S(u,v)$ can be approximated by a PDE, as formulated by Equation (1). Using the method of separation of variables, an analytic solution to Equation (1) is given:

$$S(u,\,v)=A_0(u)+\sum_{n=1}^{\infty}\left[A_n(u)\cos(nv)+B_n(u)\sin(nv)\right] \tag{2}$$

where $A_0(u)$ is considered to be the "spine" of the reconstructed surface, while the remaining terms represent a summation of "radius" vectors that give the position of reconstructed surface $S(u,v)$ relative to the "spine". As a result, the PDE surface patch may be pictured as a sum of the "spine" vector $A_0(u)$, plus a primary "radius" vector $A_1(u)\cos(v)+B_1(u)\sin(v)$, plus a secondary "radius" vector $A_2(u)\cos(2v)+B_2(u)\sin(2v)$ attached to the end of the primary "radius", and so on. The amplitude of the "radius" term decays as the frequency increases. It can be observed that the first few "radii" containing the most essential geometric information are the major contributors to surface generation while the following ones are trivial enough to be neglected. Thus, we can rewrite Equation (2) as

$$S(u,\,v)=A_0(u)+\sum_{n=1}^{N}\left[A_n(u)\cos(nv)+B_n(u)\sin(nv)\right] \tag{3}$$

where N indicates the first N "radii" with

$$A_0(u)=a_{00}+a_{01}u+a_{02}u^2+a_{03}u^3 \tag{4}$$

$$A_n(u)=a_{n1}e^{anu}+a_{n2}ue^{anu}+a_{n3}e^{-anu}+a_{n4}ue^{-anu} \tag{5}$$

$$B_n(u)=b_{n1}e^{anu}+b_{n2}ue^{anu}+b_{n3}e^{-anu}+b_{n4}ue^{-anu} \tag{6}$$

The PDE coefficients $a_{00}, a_{01}, \ldots, a_{n3}, a_{n4}$ and b_{11}, $b_{12}, \ldots, b_{n3}, b_{n4}$ are determined by Fourier-analysing the PDE boundary conditions imposed on Equation (3). The boundary conditions take the following forms:

$$S(0,\,v)=C_0(v) \tag{7}$$

$$S(u_1,\,v)=C_1(v) \tag{8}$$

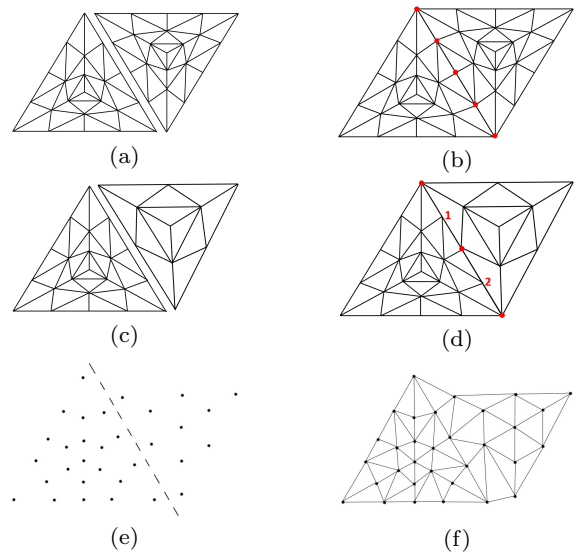


Fig. 1 Patch merging illustration of position fixers adopted in [32] and our new global triangulation scheme

$$S(u_2,\, v) = C_2(v) \tag{9}$$

$$S(1,\, v) = C_3(v) \tag{10}$$

$C_0(v)$, $C_1(v)$, $C_2(v)$, $C_3(v)$ represent the boundary conditions when $u = 0$, u_1, u_2, and 1, respectively, and $0 < u_1 < u_2 < 1$.

For each PDE patch, the number of sampling points in the uv domain determines its level of detail (LOD). Different LODs can be obtained by adjusting the granularity of the uv grid without changing the number of PDE coefficients used in generation. Generally speaking, the denser the granularity, the more details the PDE patch can retain.

3.2 Patch Merging

Since Equation (3) is used to approximate Equation (2), this approximation introduces visible distortions on the boundaries of generated PDE patches, resulting in either overlaps or seams between the patches. In order to avoid this problem, [32] introduced a group of position fixers. The use of fixers can avoid generation of T-junctions between adjacent patches and blend the patches together seamlessly.

Although the above blending scheme can achieve satisfactory results, there still exist some problems: (1) Selection of the position fixers relies on manual work, making it inefficient and inconvenient for practical applications, such as compression and filtering. (2) Position fixers need extra storage space and increase the data size in geometry representation. (3) The resolution of each PDE patch must remain identical. Fig. 1(a) shows two patches in an identical resolution and Fig. 1(b) shows the result of patch merging using the scheme in [32], where five red dots are the position fixers used. When the resolution of the patch is different (Fig. 1(c)), the merging result produced by the position fixers is undesired, as shown in Fig. 1(d), junctions '1' and '2' appear unstable.

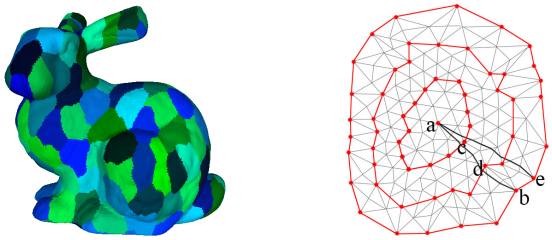
To address these problems, we propose a new merging scheme based on point cloud. Specifically speaking, the vertexes of each PDE patch are first computed using the uv grid designed in [32], but without its connectivity information. This means that in the new scheme, all the reconstructed PDE patches containing only vertexes are merged into one model in terms of point cloud. A final mesh can then be obtained by triangulating the PDE point cloud with any existing algorithm. In this paper, we employ the ball pivoting algorithm [2] in progressive mesh compression and Poisson surface reconstruction [23] in mesh denoising according to their specialities. We call the new scheme global triangulation, different from the local one previously introduced in [32]. This global triangulation scheme enables different patches with any resolutions to be seamlessly blended together, avoiding the manual intervention as well as the extra storage cost introduced by the position fixers. Fig. 1(e) and Fig. 1(f) show such an example of merging two patches in different resolutions with the new scheme. Fig. 1(e) and Fig. 1(c) have the same vertex information, but different topologies. It can be seen that the topology of Fig. 1(f) is more reasonable than that of Fig. 1(c).

4 Mesh Preprocessing

Before PDE patch generation, a mesh model needs to be preprocessed with mesh partitioning and boundary extraction. Mesh partitioning disparts the input mesh into a number of patches, each of which corresponds to one PDE. For each patch, four boundary conditions are subsequently extracted. In this section we introduce the new mesh partitioning and boundary extraction schemes.

4.1 Mesh Partitioning

In [32], mesh simplification was used to guide partitioning a mesh model, where a high-resolution triangular



(a) (b)

Fig. 2 (a) A result of 160 patches partitioned by MeTiS. (b) Illustration of the extracted boundary conditions. c and d are extracted as boundary vertexes because they are closest to the one-third and two-thirds trisecting points on the geodesic line between a and b

mesh was simplified to a user-specified resolution, and then this simplified mesh was used as reference to guide partitioning the original mesh. Thus, the PDE patches in [32] were decided by the mesh simplification approach adopted rather than the geometric mesh itself. If the simplification approach adopted gave no consideration to the geometric detail of the original shape, it would lead to a severe loss of original mesh information after PDE reconstruction. To this end, we have to find a new segmentation method.

The existing mesh segmentation approaches can be divided into part-type segmentation and surface-type segmentation [31]. Part-type segmentation is based on human perception and focuses on partitioning the mesh into meaningful 3D volumetric components, while surface-type segmentation uses surface geometric properties of the mesh, such as curvature and planarity, to dispart the mesh into surface patches. Topologically, part-type segmentation cannot guarantee segmented parts a homeomorphism, while patches segmented by surface-type methods are topologically equivalent to a disk. Therefore, a surface-type method should be adopted here due to the use of PPDE.

Ideally, we hope to partition a mesh into fewer patches for a smooth region while more patches for a detailed region. Nevertheless this ideal partitioning result is hardly achieved by the existing surface-type methods due to geometry uncertainty. Even for those geometry-aware surface-type methods, segmentation results may suffer from scale disparity, oversensitivity to local details, or even topology variation, etc. Therefore, rather than using a geometry-aware method, we resort to topology-friendly software MeTiS [22] in this paper. In [20], the effectiveness and feasibility of MeTiS have been demonstrated for a spectral mesh compression method. Compared with the simplification scheme adopted in [32], patches segmented by MeTiS retain both the geometry and topology of the original mesh and have the same number of vertices. Fig.2(a) shows the partitions generated by MeTiS for the *Stanford bunny*.

4.2 Boundary Extraction

In our method four boundary curves are extracted from each patch after MeTiS partitioning. We first calculate the average position of all the vertexes of each patch, and then select the vertex closest to the average position as the first boundary condition of this patch, i.e. the innermost boundary condition. Next, the outermost vertexes of the patch are extracted as the fourth boundary condition. In order to precisely approximate the patch with complex geometry, all the outermost vertexes of the patch are used in PDE patch generation. After obtaining the fourth boundary condition, corresponding geodesic lines between the vertexes on the first and fourth boundaries are calculated, and the second and third boundary conditions are in turn extracted by collecting the vertexes closest to the one-third and two-thirds trisecting points on the geodesic lines.

Note that our boundary extraction scheme differs from the one used in [32], which was tailored for triangle-shaped PDE patches. In [32], the corner points of each triangle-shaped patch are first located after simplifying the original mesh, and then the remaining points on the boundary curves are selected by seeking the vertexes in the original mesh closest to the triangular edges of the simplified mesh. Since a fixed number of boundary points are required, if vertexes on the original mesh are insufficient to select, then interpolation is carried out. There is one complication that the vertexes obtained by interpolation may not locate in the original mesh, resulting in some distorted details. Instead of interpolation, if the vertexes are insufficient for the second and third boundaries, the boundary extraction method in this paper allows the same vertex to be selected more than once, guaranteeing that all the boundary points originate from the mesh vertexes. Fig. 2(b) shows an example of boundary extraction of an arbitrary patch using our method, where the vertexes for the second boundary are insufficient to select. The vertex indicated by alphabet c is selected twice because it is the vertex closest to the one-third trisecting points of both the geodesic lines linking a and b, and a and e.

5 Progressive Mesh Compression

After mesh partitioning and boundary extraction, we are ready for progressive mesh compression. In progressive mesh compression a coarse 3D mesh is first transmitted in lower precision and then decompressed

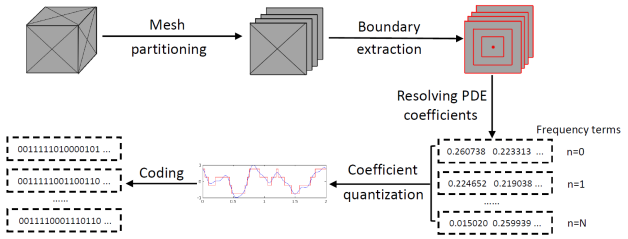


Fig. 3 The pipeline of progressive compression

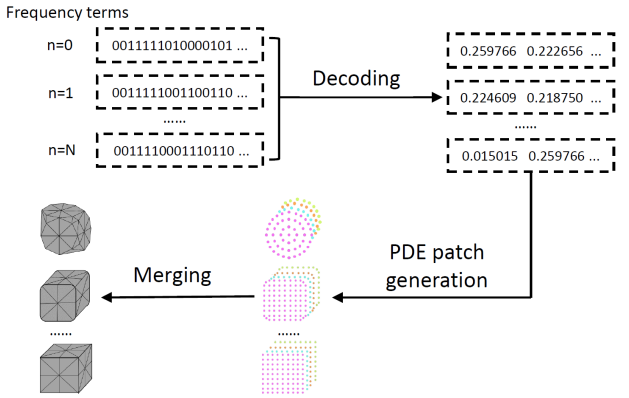


Fig. 4 The pipeline of progressive decompression

at the receiving end. With more geometric information sent from the transmitting end and subsequently decompressed at the receiving end, the coarse mesh will be gradually refined to restore the original model. In the PPDE method, the PDE coefficients are resolved by imposing Fourier-analysis on the boundary conditions. Fourier analysis copes with the mesh data in the frequency domain. According to the spectral characteristic that low-frequency components contribute more to the reconstructed mesh than the high-frequency ones, we can obtain a coarse model of the original mesh with its basic geometric structure by transmitting only the coefficients of $A_0(u)$ in Equation (3), i.e., a_{00}, a_{01}, a_{02}, and a_{03}, and then refine the coarse mesh with the received coefficients of the following N "radius" terms, i.e., $a_{11}, a_{12}, \ldots, a_{n3}, a_{n4}$ and $b_{11}, b_{12}, \ldots, b_{n3}, b_{n4}$. We divide our progressive mesh compression scheme into progressive compression and progressive decompression.

Progressive Compression: Our progressive compression procedure is composed of five stages and illustrated in Fig.3. An input mesh model is first partitioned into a number of patches with MeTiS, and for each patch four boundary conditions are then extracted. The corresponding PDE coefficients of each patch can be calculated by imposing Fourier analysis on its four boundary conditions. In this paper, each PDE coefficient is empirically quantized to 15 bits, and the total quantity of PDE coefficients is $12 \times (2N + 1)$ for each patch. Using the Lempel-Ziv-Markov chain algorithm (LZMA), we can in turn encode the PDE coefficients from low-frequency to high-frequency.

Progressive Decompression: Our progressive decompression process is composed of three stages and shown in Fig.4. During decompression, the low-frequency coefficients are first downloaded and decoded so that a coarse mesh can be reconstructed using the PPDE method. With the higher frequency coefficients received, more geometric details of the original mesh are recovered. Note that our method first renders the model into a point cloud, before this point cloud model is triangulated using the ball-pivoting algorithm [2] for its low computational complexity.

6 Mesh Denoising

Raw mesh data acquired from scanning devices inevitably contain noise, and thus, mesh denoising algorithms are required ahead of any further mesh process. In image processing, the Fourier transform has been widely used in transforming an image from its spatial domain into its frequency domain. However, the image domain is regularly sampled, while 3D geometric meshes are not. Therefore, it is difficult to directly apply the Fourier transform to mesh signal. Previously, we have demonstrated that a 3D model can be reconstructed with the PPDE method by manipulating frequency coefficients of the Fourier series expansion. Likewise, we can take advantage of the spectral nature of the PPDE method to achieve a low-pass filter by retaining the low-frequency coefficients while discarding the high-frequency ones.

Fig.5 shows the pipeline of our denoising scheme using the PPDE method. An input mesh with noise is first partitioned into a number of patches. For each noised patch, four boundary conditions are extracted and the corresponding PDE coefficients are calculated by Fourier analysis. Preceding the generation of point cloud of the PDE patches, low-pass filtering is carried out. In the frequency domain, low-frequency components correspond to the basic geometric structure of the model, while the high-frequency ones define geometric details of the model, such as bumps and noise. Thus, by keeping the PDE coefficients of first N "radius" terms while discarding those of the rest "radii", a result of low-pass filtering can be obtained after merging the patches. Our scheme also allows for iterative filtering if the denoising result of one iteration is unsatisfactory.

Note that instead of using the ball pivoting algorithm, we adopt the Poisson surface reconstruction algorithm for triangulation mainly because of the smooth-

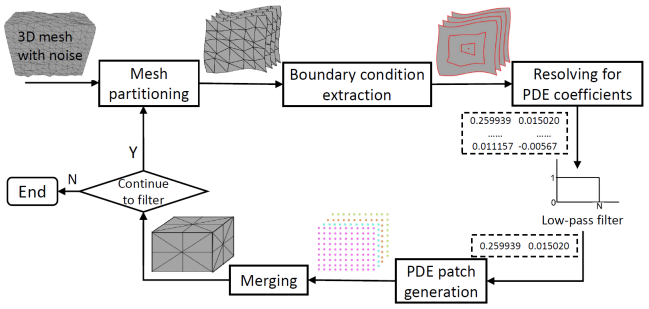


Fig. 5 The pipeline of our denoising scheme

ness it lends, which makes the Poisson algorithm ideal for low-pass filtering.

7 Experimental Results

In this section, we experimentally assess the performance of our methods introduced in this paper against the existing methods whose codes are publicly available. Fig.6 shows five typical mesh models tested in our experiments, featuring various characteristics including smooth surface as well as sharp edges, *etc.*

7.1 Progressive Mesh Compression Evaluations

We assess our compression method by comparing it with one spatial and one spectral progressive mesh compression methods. Fig.7-11 shows the rate-distortion curves of the Wavemesh algorithm [38], RDO algorithm [26], and ours for the testing mesh models, where the horizontal axes indicate compression rates while the vertical axes indicate the corresponding Root Mean Square (RMS) errors evaluated by METRO [9], measuring how close a reconstructed mesh is to the original. When the compression rate keeps unchanged, the smaller the RMS error, the better the compression performance. When the RMS error remains the same, the lower the compression rate, the better the compression performance. All the corresponding coefficients and vertexes in these three algorithms are empirically quantized with 15 bits.

In our method, there are three variables, N, M, and L, to be decided beforehand. N determines the number of PDE coefficients used during reconstruction; M denotes the number of patches partitioned by MeTiS; L indicates the number of isoparm triangle layers in the uv grid [32], determining the LOD of reconstructed patches.

For the sake of visibility, we compare the three algorithms within a specified range in Fig.7-11. The nodes, from left to right, of the polylines of our scheme in

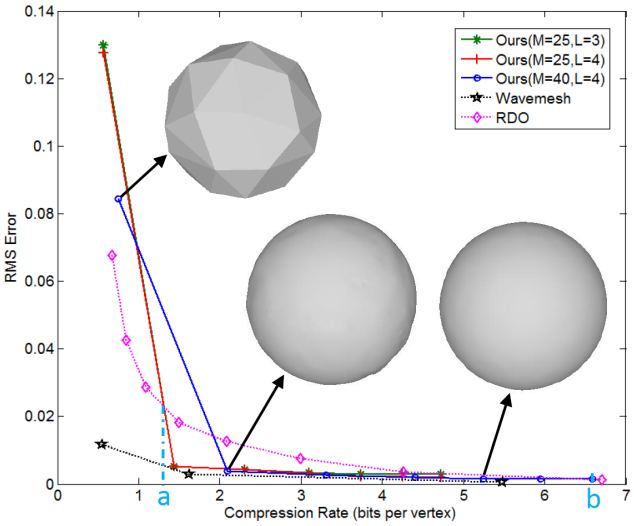


Fig. 7 The rate-distortion curves of the sphere model

Fig.7-11 correspond to the results when the value of N increases from 0 to 6. It can be observed that the value of N determines how close the reconstructed mesh is to the original one. The larger the value of N, the smaller the RMS error, the closer the reconstructed mesh to the original. However, the increase of N will enlarge the compression rate. Note that for the sake of comparison, we visualize all the possible nodes generated by the two competing algorithms within the specified ranges in Fig.7-11, which correspond to different levels of detail with variant vertexes.

The patch number M also influences the compression results. It can be observed in Fig.7-11 that with N fixed, the larger the value of M, the smaller the RMS error, i.e. the more details the reconstructed mesh can preserve, but this may lead to an increase in compression rate. Therefore, in practical use we should choose an appropriate M for a trade-off between the compression rate and RMS error.

In addition, the resolution of the uv parametric grid also influences the reconstruction precision. It can be observed in Fig.7-11 that with both M and N fixed, the larger the value of L, the smaller the RMS error, but the compression rates remain unchanged.

As can be seen, our method achieves better performances than the RDO algorithm between the ranges indicated by the alphabets a and b in Fig.7-10. Our method can also achieve better performances than the Wavemesh algorithm between the indicated ranges in Fig.8-10. In Fig.11, both the Wavemesh and RDO algorithms perform better than ours because the partitioning method used in this paper is insensitive to the model with sharp edges.

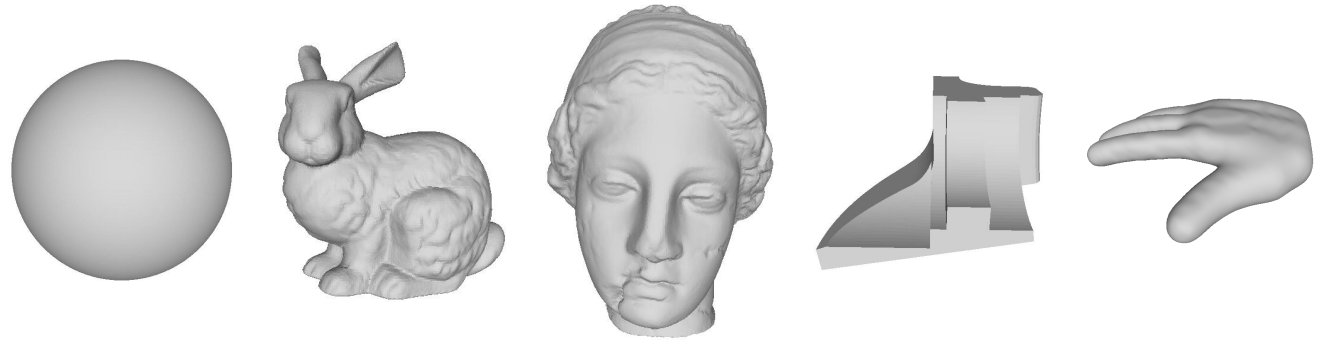


Fig. 6 The mesh models used in our experiments. From left to right are the sphere model, *Stanford bunny*, *Venus*, *Fandisk*, and hand model

We also observe from Fig.7-11 that when the compression rate is greater than b, the RMS error keeps decreasing for the Wavemesh and RDO algorithms, but almost unchanged for our scheme. This is because both the Wavemesh and RDO algorithms are lossless, and more geometric details can be recovered by the two algorithms as more bits are used. By contrast, the impact of high-frequency components after some "radius" terms in our PDE method is too weak to be observed.

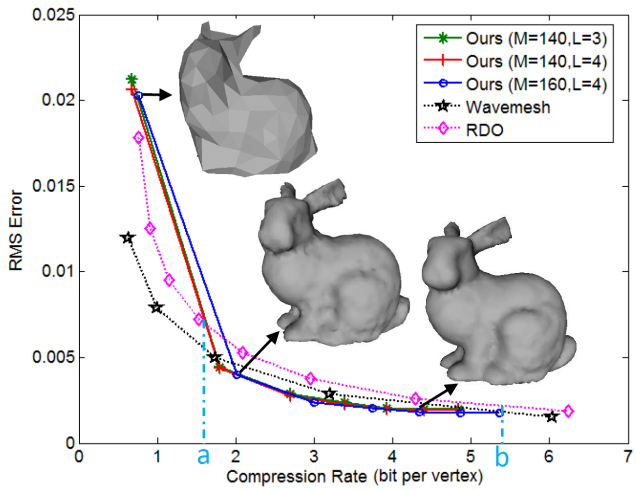


Fig. 8 The rate-distortion curves of the Bunny

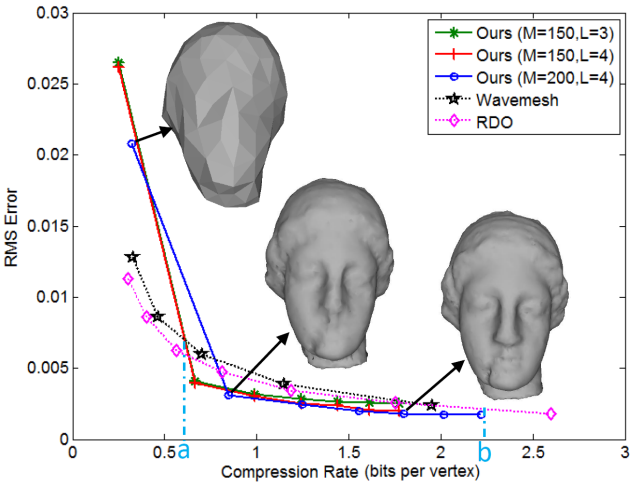


Fig. 9 The rate-distortion curves of Venus

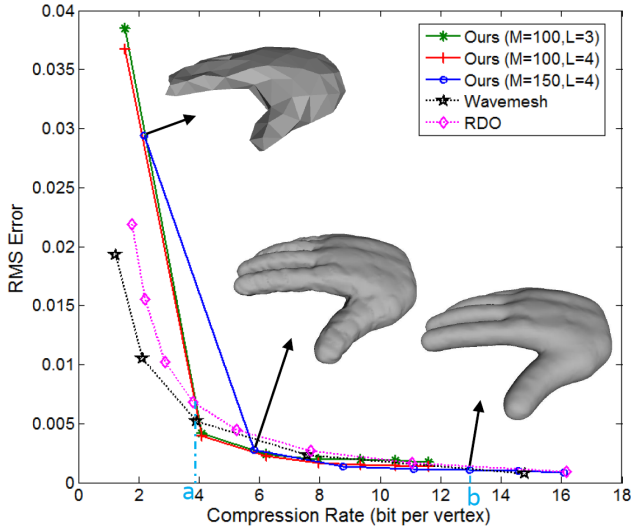


Fig. 10 The rate-distortion curves of the hand model

7.2 Mesh Denoising Evaluations

In this section, we compare our denoising scheme with some existing methods, such as FDCO [13], JDD [19], SRML [34], local ZFAT [44], ZDZBL [43], and LCS [16]. Each of the above methods involves a set of parameters to be set by the user, and the best parameters may vary from model to model. For a fair comparison, we choose the best result for each method by sampling parameters. Similarly, we need to set parameters properly in

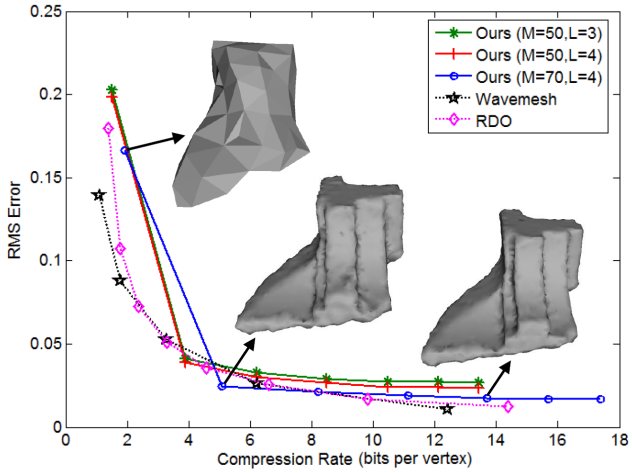


Fig. 11 The rate-distortion curves of the Fandisk

our scheme to produce desired results. In our scheme, we are concerned with four parameters, M, N, L, and I, where I indicates the number of filtering iterations.

In order to test the denoising effect, Gaussian noise is applied to both mesh normal and random directions. As same as [43], the intensity of Gaussian noise in this paper is controlled by a relative variance parameter, defined as

$$\sigma_E = \frac{\sigma}{E_{mean}} \tag{11}$$

where σ is the variance of the Gaussian function, and E_{mean} is the average edge length of a mesh.

We first evaluate the robustness of variant denoising methods against Gaussian noise with an increasingly growing intensity, as shown in Fig.12. The *Stanford bunny* is degraded by adding Gaussian noise to its vertexes along the vertex normals. When the intensity of noise is low, all the seven denoising algorithms can achieve desired subjective results. However, as the noise intensity increases, the denoising results of the FDCO, JDD, SRML and ZFAT algorithms become worse and worse, while ZDZBL, LCS and our algorithms can shake off the impact of the growing noise intensity. The parameter values of our algorithm adopted in Fig.12 are listed in Table 1.

Table 2 shows the iteration numbers performed by the seven competing denoising algorithms in Fig.12, where n_{iter} denotes the number of normal iterations, and v_{iter} indicates the number of vertex iterations. It is observed that to depress the impact of the growing noise intensity the five competing algorithms have had to increase the iteration number, except the JDD algorithm that is independent of iteration. By contrast, our algorithm can generate desired denoising results with only two iterations.

Table 1 The parameters adopted by our algorithm in Fig.12

The intensity of noise σ_E	Parameters			
	M	N	L	I
0.2	300	5	3	2
0.4	300	6	3	2
0.6	300	6	3	2
0.8	300	6	3	2

Table 2 The iterations required by variant algorithms in Fig.12

σ_E	Algorithms	Iterations	
0.2	[FDCO]	40	
	[JDD]	-	
	[SRML]	n_{iter}	v_{iter}
		5	20
	[ZFAT]	n_{iter}	v_{iter}
		6	20
	[ZDZBL]	n_{iter}	v_{iter}
		4	15
	[LCS]	n_{iter}	v_{iter}
		5	10
	Ours	**2**	
0.4	[FDCO]	40	
	[JDD]	-	
	[SRML]	n_{iter}	v_{iter}
		30	50
	[ZFAT]	n_{iter}	v_{iter}
		10	20
	[ZDZBL]	n_{iter}	v_{iter}
		4	15
	[LCS]	n_{iter}	v_{iter}
		7	15
	Ours	**2**	
0.6	[FDCO]	80	
	[JDD]	-	
	[SRML]	n_{iter}	v_{iter}
		50	50
	[ZFAT]	n_{iter}	v_{iter}
		15	40
	[ZDZBL]	n_{iter}	v_{iter}
		4	15
	[LCS]	n_{iter}	v_{iter}
		10	15
	Ours	**2**	
0.8	[FDCO]	50	
	[JDD]	-	
	[SRML]	n_{iter}	v_{iter}
		100	100
	[ZFAT]	n_{iter}	v_{iter}
		20	50
	[ZDZBL]	n_{iter}	v_{iter}
		20	10
	[LCS]	n_{iter}	v_{iter}
		10	25
	Ours	**2**	

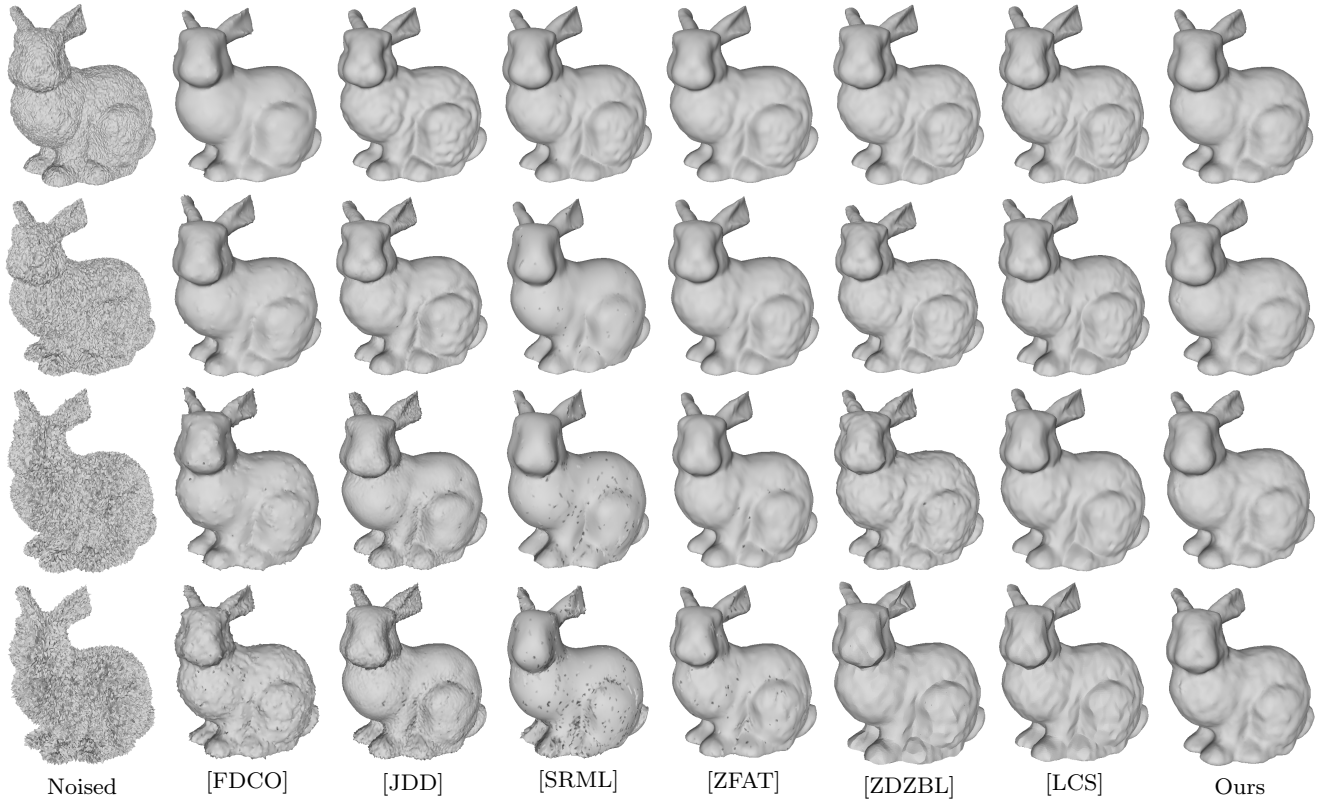


Fig. 12 A subjective comparison of the denoising algorithms for the *Stanford bunny* with additive Gaussian noise of various intensities along the normal directions. The intensity σ_E of noise from top to bottom are in turn set to 0.2, 0.4, 0.6, and 0.8

Fig.13 and Fig.14 show denoising performance comparisons between our algorithm and the competing ones on variant models whose geometric details from top to bottom become sharper and sharper. Especially, the model *Fandisk* contains clear edges. Fig.13 shows the results for the Gaussian noise added along the normal directions. For the sphere, *Venus*, and hand models all the seven algorithms can achieve subjectively accepted denoising results. For the *Fandisk*, the FDCO and JDD algorithms cannot completely filter out the noise; the SRML and ZFAT algorithms produce incorrect normals; our algorithm fails to reserve the sharp edges mainly because our spectral method can hardly distinguish noise and sharp geometric detail and MeTiS is insensitive to geometry; ZDZBL and LCS algorithms can achieve desired subjective results. Fig.14 shows the denoising results for the Gaussian noise added along random directions. As can be seen, the FDCO algorithm does not completely filter out the noise for all the mesh models. The other six denoising algorithms can produce subjectively accepted denoising results for the first three mesh models, but not for the *Fandisk* and hand model except the LCS algorithm. JDD, SRML, ZFAT and ZDZBL algorithms produce incorrect normals; our algorithm can produce desired denoising result for the hand model, but not for the *Fandisk*. Note that since our denoising scheme relies on discarding high-frequency components of the mesh, some geometric detail will be inevitably removed along with noise. Such impacts can be seen in the face of the *Venus*, as well as the edges of the *Fandisk*. The parameters of our algorithm adopted in Fig.13 and Fig.14 are listed in Table 3.

Fig. 15 shows a further comparison of denoising algorithms on the models with a higher level of noise added along random directions. As can be seen, FDCO, JDD, SRML, ZFAT, and ZDZBL algorithms fail to achieve accepted denoising results for all the mesh models. LCS and our algorithms can produce subjectively accepted denosing results for all the mesh models, except our algorithm fails to reserve the sharp edges of the *Fandisk*.

Since error metrics, such as the RMS, max, mean, and Hausdorff distances are based on correspondences of vertexes, edges, and similar triangles faces, and there is no such one-to-one correspondence in our method between the denoised and the original, an objective assessment for our method may not reflect the reality. For instance, in Fig.14, the denoising result of our algorithm for *Venus* is visually better than that of FDCO, and FDCO even fails to filter out the noise of *Venus*.

Table 3 The parameters adopted by our algorithm in Fig.13 and Fig.14

Figures	Models	Parameters			
		M	N	L	I
Fig.13	Sphere	500	1	2	4
	Venus	500	6	3	1
	Fandisk	200	2	2	2
	Hand	300	3	3	2
Fig.14	Sphere	500	1	2	3
	Bunny	400	6	3	1
	Venus	500	6	4	1
	Fandisk	200	5	2	2
	Hand	300	5	3	2

Nonetheless, the RMS, max, mean, and Hausdorff distances of FDCO for *Venus* are smaller than ours, which is, however, against the reality. Therefore, in this paper the denoising results are not assessed by the error metrics.

8 Concluding Remarks

The feasibility of the PDE method in 3D mesh signal processing is explored for the first time in this paper. To accommodate progressive mesh compression and mesh denoising, we upgrade the existing PPDE method in patch merging, mesh partitioning, and boundary extraction. Although our primary goal is not to develop a progressive mesh compression algorithm or mesh denoising algorithm excelling the extant ones, the experiments have demonstrated the advantages of our method to some extent.

Our denoising algorithm is performed in the spectral domain. One disadvantage of spectral methods is that they can hardly distinguish noise and geometric details of a model in the spectral domain. Thus the spectral denoising method will inevitably trade off some geometric details of the original mesh during denoising, resulting in over-smoothing. For example, in both Fig.13 and Fig.14, our denoising algorithm fails to restore some details in the face of the *Venus*, nor the sharp edges of the *Fandisk*.

This paper also opens some windows for further research. For example, in progressive mesh compression, apart from the quantity of frequency terms, the patch number M also influences the compression rate and reconstruction precision of our method. This means that to achieve a small compression rate we can partition the mesh into a small number of patches, but this may lead to unprecise reconstruction. However, we need to seek a trade-off between the compression rate and RMS error by properly selecting an optimal M, and to seek a way of replacing manual setting, which are yet to be studied in the future. Moreover, we adopt topology-friendly MeTiS in this paper to segment the mesh model into patches with an identical number of vertexes, which is, however, geometry-insensitive. Since desirable properties of a surface-type segmentation method may vary according to the specific application, it would be better in the future to develop our own geometry-aware segmentation algorithm for this specific application, which should enable a planar region to possess larger patches and a detailed region to possess smaller patches, and meanwhile, should be robust enough against scale disparity, oversensitivity, and topology variation that the existing surface-type geometry aware algorithms are facing.

Note that both our progressive mesh compression scheme and mesh denoising scheme currently work only for genus-0 models. This is determined by our patch merging scheme, where the point cloud is directly triangulated to achieve seamless blending of PDE patches.

Acknowledgement

The authors would like to thank the reviewers for their constructive comments. The authors would also like to thank Xuequan Lu for his quick and responsible assistance during the revision of this paper. This work has been supported by the National Natural Science Foundation of China (61202291).

References

1. Ahmat, N., Ugail, H., Castro, G.G.: Method of modelling the compaction behaviour of cylindrical pharmaceutical tablets. International journal of pharmaceutics. 405(1), 113-121 (2011)
2. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics. 5(4), 349-359 (1999)
3. Bloor, M., Wilson, M.: Generating blend surfaces using partial differential equations. Computer-Aided Design. 21(3), 165-171 (1989)
4. Bloor, M.I., Wilson, M.J.: Using partial differential equations to generate free-form surfaces. Computer-Aided Design. 22(4), 202-212 (1990)
5. Bloor, M.I., Wilson, M.J.: Efficient parametrization of generic aircraft geometry. Journal of Aircraft. 32(6), 1269-1275 (1995)
6. Castro, G.G., Ugail, H.: Shape morphing of complex geometries using partial differential equations. Journal of Multimedia. 2(6), 15-25 (2007)
7. Chen, C., Sheng, Y., Li, F., Zhang, G., Ugail, H.: A PDE-based head visualization method with ct data. Computer Animation and Virtual Worlds. 28(1) (2017)
8. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3D mesh segmentation. ACM Transactions on Graphics. 28(3), 73 (2009)

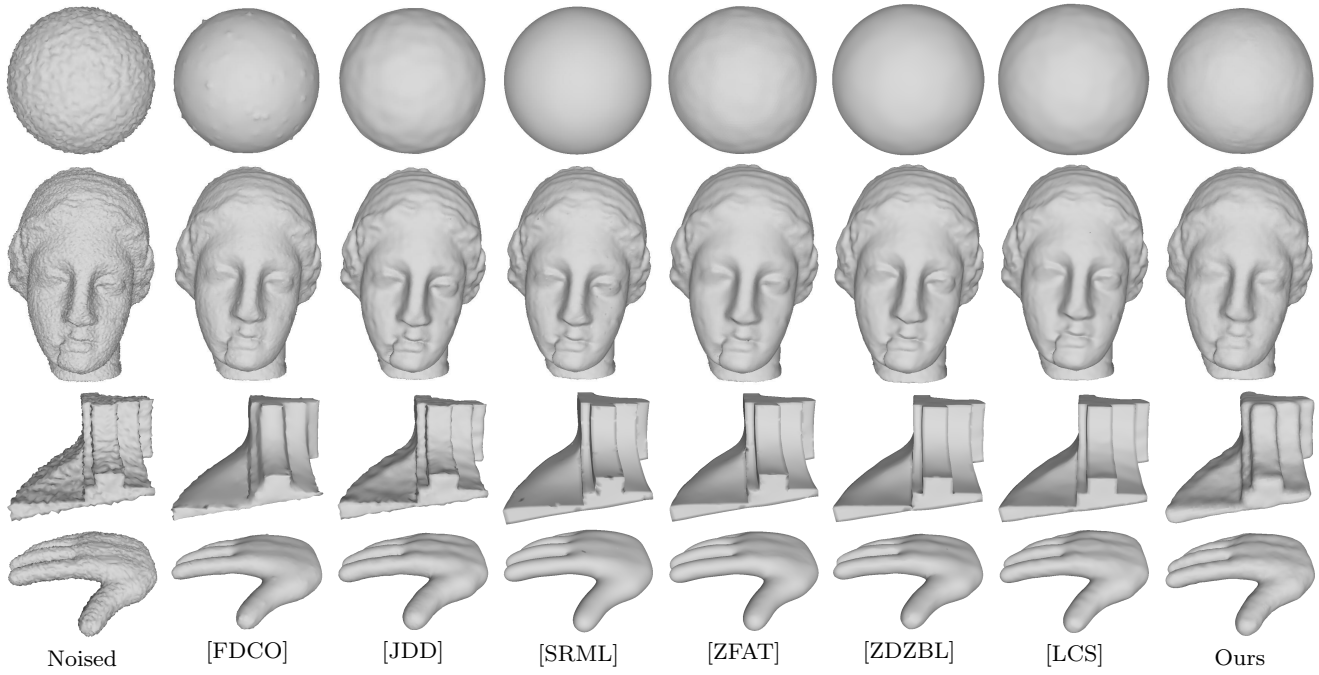


Fig. 13 A comparison of the denoising algorithms on variant meshes with additive Gaussian noise along the normal directions. The intensity σ_E of noise is set to 0.3

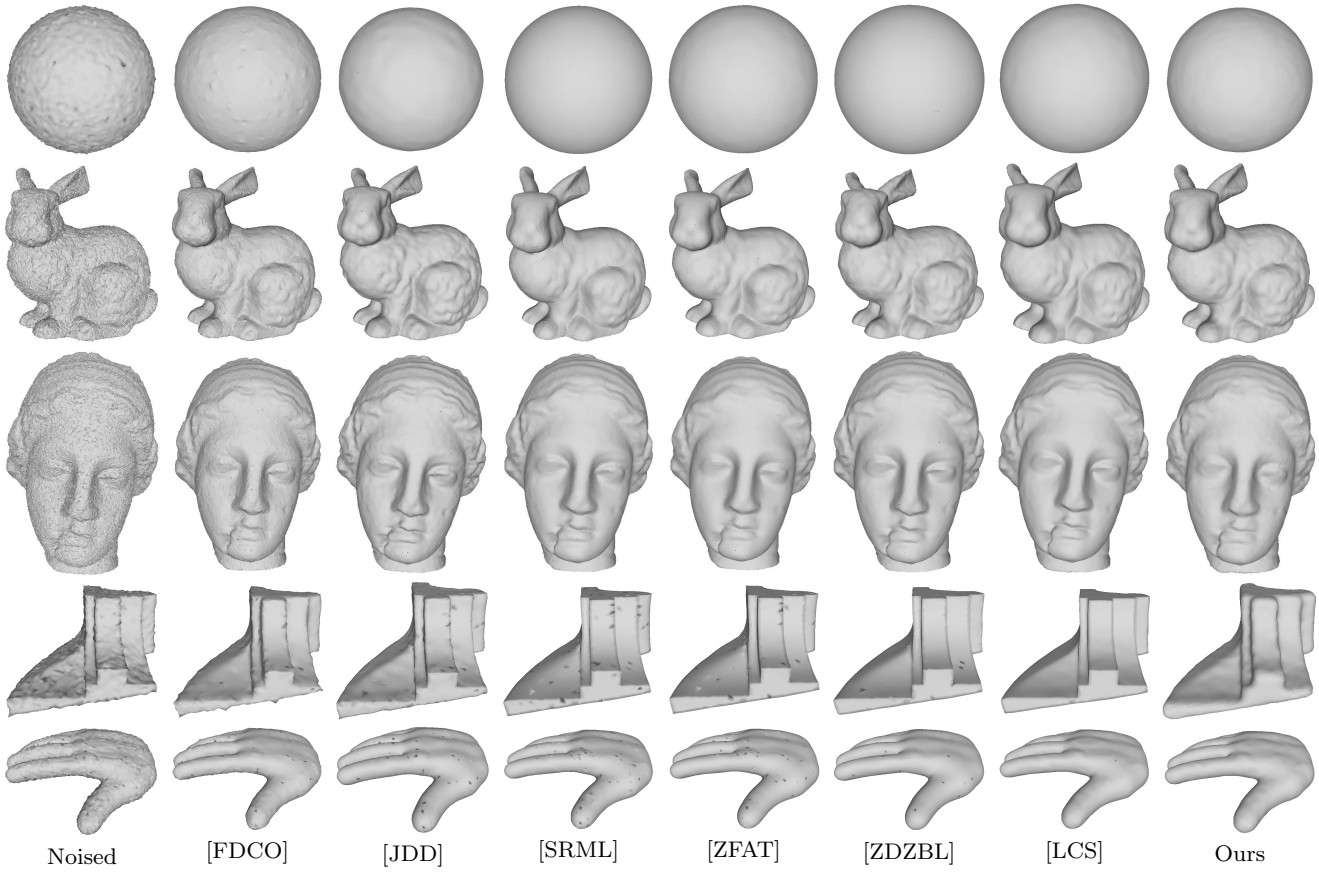


Fig. 14 A comparison of the denoising algorithms on variant meshes with additive Gaussian noise along random directions. The intensity σ_E of noise is also set to 0.3

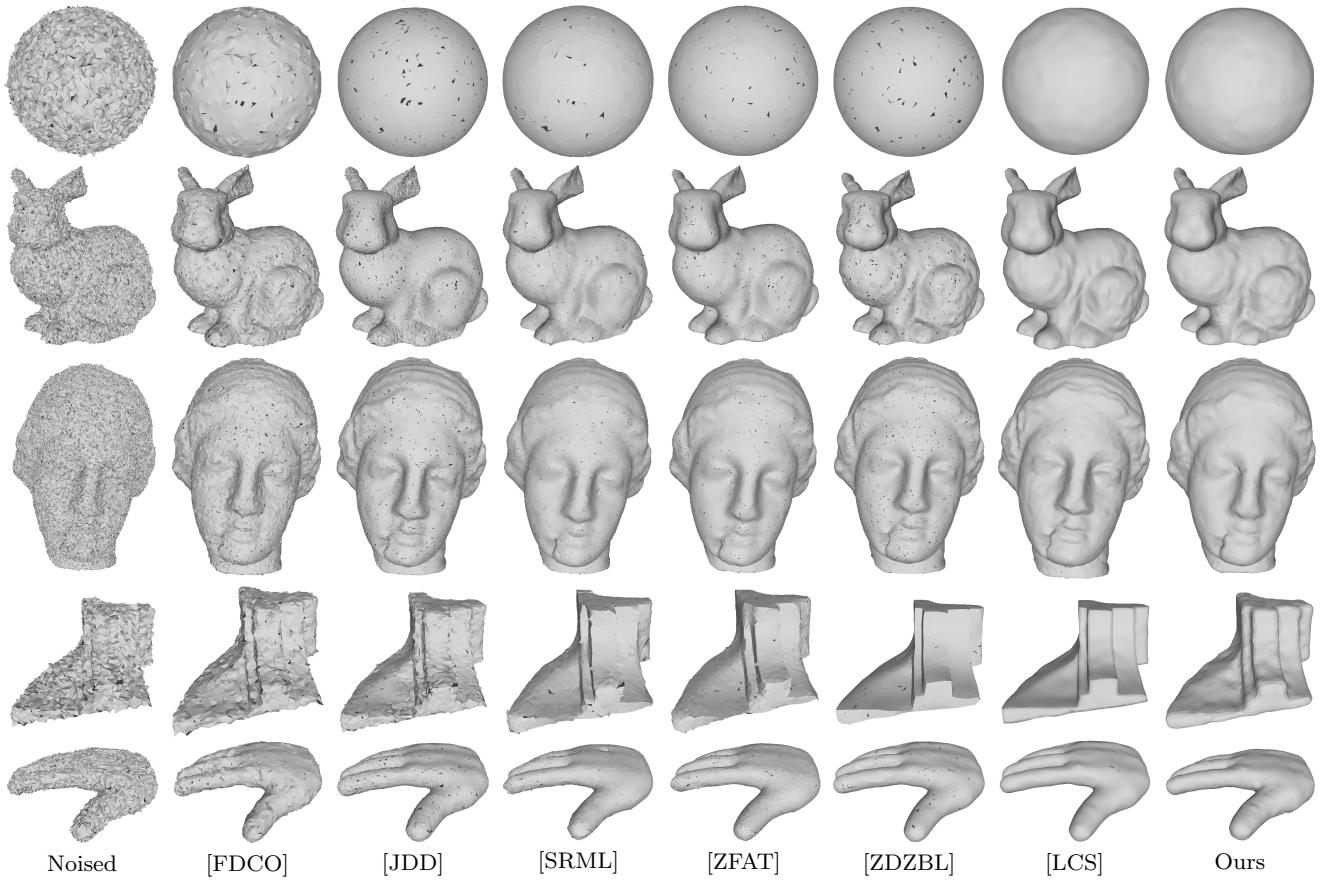


Fig. 15 A comparison of the denoising algorithms on the meshes with additive Gaussian noise along random directions. The intensity σ_E of noise is set to 0.7

9. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Computer Graphics Forum. 17(2), 167-174 (1998)
10. Dekanski, C., G. Bloor, M., Wilson, M.: Partial differential equation surface generation and functional shape optimization of a swirl port. Journal of propulsion and power. 13(3), 398-403 (1997)
11. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. Proceedings of SIGGRAPH'99. pp. 317-324 (1999)
12. Du, H., Qin, H.: Direct manipulation and interactive sculpting of PDE surfaces. Computer Graphics Forum. 19(3), 261-270 (2000)
13. Fleishman, S., Drori, I., Cohen-Or, D.: Bilateral mesh denoising. ACM Transactions on Graphics. 22(3), 950-953 (2003)
14. Gandoin, P.M., Devillers, O.: Progressive lossless compression of arbitrary simplicial complexes. ACM Transactions on Graphics. 21(3), 372-379 (2002)
15. Lu X, Deng Z, Chen W.: A robust scheme for feature-preserving mesh denoising. IEEE Transactions on Visualization and Computer Graphics. 22(3), 1181-1194 (2016).
16. Lu X, Chen W, Schaefer S.: Robust mesh denoising via vertex pre-filtering and L_1-median normal filtering. Computer Aided Geometric Design. 54, 49-60 (2017).
17. He, L., Schaefer, S.: Mesh denoising via L_0 minimization. ACM Transactions on Graphics. 32(4), 64 (2013)
18. Hoppe, H.: Progressive meshes. Proceedings of SIGGRAPH'96. pp. 99-108 (1996)
19. Jones, T.R., Durand, F., Desbrun, M.: Non-iterative, feature-preserving mesh smoothing. ACM Transactions on Graphics. 22(3), 943-949 (2003)
20. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. Proceedings of SIGGRAPH'00. pp. 279-286 (2000)
21. Karni, Z., Gotsman, C.: 3D mesh compression using fixed spectral bases. Graphics Interface. 1, 1-8 (2001)
22. Karypis, G., Kumar, V.: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN (1998)
23. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Transactions on Graphics. 32(3), 29 (2013)
24. Kim, B., Rossignac, J.: Geofilter: Geometric selection of mesh filter parameters. Computer Graphics Forum. 24(3), 295-302 (2005)
25. Kubiesa, S., Ugail, H., Wilson, M.: Interactive design using higher order PDEs. The Visual Computer. 20(10), 682-693 (2004)
26. Lee, H., Lavoué, G., Dupont, F.: Rate-distortion optimization for progressive compression of 3D mesh with color attributes. The Visual Computer. 28(2), 137-153 (2012)
27. Pajarola, R., Rossignac, J.: Compressed progressive meshes. IEEE Transactions on Visualization and Computer Graphics. 6(1), 79-93 (2000)
28. Pang, M.Y., Sheng, Y., Sourin, A., Castro, G.G., Ugail, H.: Automatic reconstruction and web visualization of com-

plex PDE shapes. International Conference on Cyberworlds. pp. 97-104 (2010)
29. Pauly, M., Gross, M.: Spectral processing of point-sampled geometry. Proceedings of SIGGRAPH'01. pp. 379-386. ACM (2001)
30. Peng, J., Kuo, C.C.J.: Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. ACM Transactions on Graphics. 24(3), 609-616 (2005)
31. Shamir A.: A survey on mesh segmentation techniques. Computer graphics forum. 27(6), 1539-1556 (2008).
32. Sheng, Y., Sourin, A., Castro, G.G., Ugail, H.: A PDE method for patchwise approximation of large polygon meshes. The Visual Computer. 26(6), 975-984 (2010)
33. Sheng, Y., Willis, P., Castro, G.G., Ugail, H.: Facial geometry parameterisation based on partial differential equations. Mathematical and Computer Modelling. 54(5), 1536-1548 (2011)
34. Sun, X., Rosin, P., Martin, R., Langbein, F.: Fast and effective feature-preserving mesh denoising. IEEE Transactions on Visualization and Computer Graphics. 13(5), 925-938 (2007)
35. Taubin, G.: A signal processing approach to fair surface design. Proceedings of SIGGRAPH'95. pp. 351-358. ACM (1995)
36. Taubin, G., Guéziec, A., Horn, W., Lazarus, F.: Progressive forest split compression. Proceedings of SIGGRAPH'98. pp. 123-132 (1998)
37. Ugail, H.: Generalized partial differential equations for interactive design. International Journal of Shape Modeling. 13(02), 201-223 (2007)
38. Valette, S., Prost, R.: Wavelet-based progressive compression scheme for triangle meshes: Wavemesh. IEEE Transactions on Visualization and Computer Graphics. 10(2), 123-129 (2004)
39. You, L., Zhang, J.J., Comninos, P.: Generating blending surfaces with a pseudo-levy series solution to fourth order partial differential equations. Computing. 71(4), 353-373 (2003)
40. Zhang, H., Fiume, E.: Butterworth filtering and implicit fairing of irregular meshes. Proceedings of Pacific Graphics. pp. 502-506 (2003)
41. Zhang, J., You, L., et al.: Fast surface modelling using a 6th order PDE. Computer Graphics Forum. 23(3), 311–320 (2004)
42. Zhang, J.J., You, L.: PDE based surface representation-vase design. Computers & Graphics. 26(1), 89-98 (2002)
43. Zhang, W., Deng, B., Zhang, J., Bouaziz, S., Liu, L.: Guided mesh normal filtering. Computer Graphics Forum. 34(7), 23-34 (2015)
44. Zheng, Y., Fu, H., Au, O.K.C., Tai, C.L.: Bilateral normal filtering for mesh denoising. IEEE Transactions on Visualization and Computer Graphics. 17(10), 1521-1530 (2011)