Block based Spectral Processing of Dense 3D Meshes using Orthogonal Iterations

Aris S. Lalos, Gerasimos Arvanitis, Anastasios Dimas and Konstantinos Moustakas
Department of Electrical and Computer Engineering, University of Patras, Patras, Greece

Keywords: Graph Signal Processing, Mesh Compression, Mesh Denoising.

Abstract: Spectral methods are widely used in geometry processing of 3D models. They rely on the projection of the mesh geometry on the basis defined by the eigenvectors of the graph Laplacian operator, becoming computationally prohibitive as the density of the models increases. In this paper, we propose a novel approach for supporting fast and efficient spectral processing of dense 3D meshes, ideally suited for real time compression and denoising scenarios. To achieve that, we apply the problem of tracking graph Laplacian eigenspaces via orthogonal iterations, exploiting potential spectral coherences between adjacent parts. To avoid perceptual distortions when a fixed number of eigenvectors is used for all the individual parts, we propose a flexible solution that automatically identifies the optimal subspace size for satisfying a given reconstruction quality constraint. Extensive simulations carried out with different 3D meshes in compression and denoising setups, showed that the proposed schemes are very fast alternatives of SVD based spectral processing while achieving at the same time similar or even better reconstruction quality. More importantly, the proposed approach can be employed by several other state of the art denoising methods as a preprocessing step, optimizing both their reconstruction quality and their computational complexity.

1 INTRODUCTION

In recent years, there has been increasing interest from researchers, system designers, and application developers on acquiring, processing, transmitting and storing 3D models, facilitating several real time applications, e.g., mobile cloud gaming (Cai et al., 2013) and 3D Tele-immersion (Alexiadis et al., 2013; Mekuria et al., 2014). These models usually come as very large and noisy meshes that stand in need of solutions for a diversity of problems including mesh compression, smoothing, symmetry detection, watermarking, surface reconstruction, and re-meshing (Zhang et al., 2010). Spectral methods have been developed with the intention of solving such problems by manipulating the eigenvalues, eigenvectors, eigenspace projections, or a combination of these, derived from the graph Laplacian operator. The processing and memory requirements of these methods are strongly dependent on the number of vertices of the 3D model, and therefore become prohibitive as the vertex density increases, especially in cases where the models are too large and need to be canned in parts, generating a sequence of 3D surfaces that arrive sequentially in time. To address this issue, the raw geometry data could be divided and processed in blocks that represent the different parts of a mesh (submeshes), as suggested in (Lalos et al., 2017; Lalos et al., 2015).

The application of direct singular value decomposition (SVD) on the graph Laplacian of each submesh, requires $O\left(n_d^3\right)$ operations, where n_d is the number of vertices in a submesh. This excessively high computational complexity needed by SVD motivated us to seek for an efficient subspace tracking implementation that processes the raw geometry data in blocks and readjust only a small number of spectral coefficients of a submesh based on the corresponding spectral values of a previous submesh. The proposed approach, is based on a numerical analysis method known as orthogonal iterations (OI) (Zhang, 2009), which is capable of estimating iteratively the subspaces of interest. The speed-up is attributed to the fact that the proposed approach requires $O\left(n_d c^2\right)$ floating point operations where c is the number of spectral components utilized and $c << n_d$. Additionally, we developed a dynamic OI approach that estimates automatically the ideal c for a predefined reconstruction quality. Extensive simulations carried out with different 3D meshes in a compression and denoising setup, proved that the proposed framework is a very fast alternative of the SVD based graph Lapla-

Lalos, A., Arvanitis, G., Dimas, A. and Moustakas, K.
Block based Spectral Processing of Dense 3D Meshes using Orthogonal Iterations.
DOI: 10.5220/0006611401220132
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 1: GRAPP*, pages 122-132
ISBN: 978-989-758-287-5

cian processing methods, without introducing noticeable reconstruction errors.

The rest of the article is organized as follows. Sec. II provides a review of prior art on spectral methods and their applications in a diversity of problems. Basic definitions related to graph spectral processing of 3D meshes are provided in Sec. III. Sec. IV presents the proposed fast spectral processing approach that is based on OI. Section V provides a flexible solution that automatically identifies the optimal subspace size c that satisfies a specific reconstruction quality criterion. Section VI presents a compression and a denoising case study, where the proposed method can be effectively adopted. In Section VII, the performance of the proposed system is evaluated, by taking into account different CAD and scanned 3D models. The article is finally wrapped up with a few open research directions in Sec. VIII.

2 RELATED WORKS

Spectral methods have been used in many different computer science fields ranging from signal processing, graph theory, computer vision and machine learning. Spectral mesh processing have been inspired by all the relevant developments in the aforementioned fields. Several surveys that cover basic definitions and applications of the graph spectral methods have been introduced by Gotsman (Gotsman, 2003), Levy (Lévy, 2006), Sorkine (Sorkine, 2005) and more recently by Zhang et al. (Zhang et al., 2010). All these surveys classify the spectral methods according to several criteria related to the employed operators, the application domains and the dimensionality of the spectral embeddings used.

Graph spectral processing of 3D meshes rely on the singular/eigen-vectors and/ or eigenspace projections derived from appropriately defined mesh operators, while it has been applied in several tasks, such as, implicit mesh fairing (Kim and Rossignac, 2005), geometry compression (Sorkine, 2005; Karni and Gotsman, 2000) and mesh watermarking (Ohbuchi et al., 2001). Taubin (Taubin, 1995) first treated the mesh vertex coordinates as a 3D signal and introduced the use of graph Laplacian operators for discrete geometry processing. This analysis was motivated by the similarities between the spectral analysis with respect to mesh Laplacian and the classical Fourier analysis. A summary of the mesh filtering approaches that can be efficiently carried out in the spatial domain using convolution approaches is given by Taubin in (Taubin, 2000). Despite their applicability in a wide range of applications such as, mesh denoising, geometry compression and watermarking, they require explicit eigenvector computations making them prohibitive for real time scenarios such as streaming and content creation applications, where large 3D models are generated in parts, providing a sequence of 3D surfaces that need to be processed fast and sequentially in time.

Computing the truncated singular value decomposition, can be extremely memory-demanding and time-consuming. To overcome this limitations, *subspace tracking* algorithms have been proposed as fast alternatives relying on the execution of iterative schemes for evaluating the desired eigenvectors per incoming block of floating point data corresponding in our case, to different surface patches (Comon and Golub, 1990). The most widely adopted subspace tracking method is Orthogonal iterations (OI), due to the fact that results in very fast solutions when the initial input subspace is close to the subspace of interest, as well as the size of the subspace remains at small levels (Saad, 2016). The fact that both matrix multiplications and QR factorizations have been highly optimized for maximum efficiency on modern serial and parallel architectures, makes the OI approach more attractive for real time applications. To the best of our knowledge, subspace tracking algorithms have never been applied for graph spectral mesh processing, despite their wide success on a large range of filtering applications.

3 SPECTRAL PROCESSING OF 3D MESHES

In this work we focus on polygon models whose surface is represented using triangles. Let us assume that each triangle mesh $\mathcal{M}$ with n vertices can be represented by two different sets $\mathcal{M} = (V,F)$ corresponding to the vertices (V) that represent the geometry information and the indexed faces (F) of the mesh. Each vertex can be represented as a point $\mathbf{v}_i = (x_i, y_i, z_i)\ \forall\ i = 1,n$ and each centroid of a face as $\mathbf{m}_i = (\mathbf{v}_{i1} + \mathbf{v}_{i2} + \mathbf{v}_{i3})/3\ \forall\ i = 1,l$. A set of edges ($E$) can be directly derived from V and F, which correspond to the connectivity information.

Spectral processing approaches, e.g., (Sorkine, 2005; Karni and Gotsman, 2000) are based on the fact that smooth geometries should yield spectra, dominated by low frequency components and suggest projecting the Cartesian coordinates $\mathbf{x},\mathbf{y},\mathbf{z} \in \Re^{n\times 1}$ in the basis spanned by the eigenvectors $\mathbf{u}_i$, $i = 1,...,c << n$ of the Laplacian operator $\mathbf{L}$ that is calculated as follows:

$$\mathbf{L} \quad = \quad \mathbf{D} - \mathbf{C}, \tag{1}$$

where $\mathbf{C} \in \Re^{n \times n}$ is the weighted connectivity matrix of the mesh with elements:

$$\mathbf{C}_{(i,j)} = \begin{cases} \frac{1}{\|\mathbf{v}_i - \mathbf{v}_j\|_2^2} & (i,j) \in (E) \\ 0 & otherwise, \end{cases}, \quad (2)$$

matrix $\mathbf{D}$ is the diagonal matrix with $\mathbf{D}_{(i,i)} = |N(i)|$, and $N(i) = \{j \,|\, (i,j) \in (E)\}$ is a set of the immediate neighbors for node i. The weighted adjacency matrix is ideal for emphasizing the coherence between Laplacian matrices of different submeshes by providing geometric information; on the contrary, the binary provides only connectivity information. Eigenvalue decomposition of $\mathbf{L}$ is written as:

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T \quad (3)$$

where Λ is a diagonal matrix consisting of the eigenvalues of $\mathbf{L}$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$ is the matrix with the eigenvectors $\mathbf{u}_i \in \Re^{n \times 1}$ which is needed to generate the spectral coefficients that are essential in providing sparse representations of the raw geometry data (Sorkine, 2005).

Similar to classical Fourier transform, the eigenvectors and eigenvalues of the Laplacian matrix L provide a spectral interpretation of the 3D signal. The eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ can be considered as graph frequencies, and the eigenvectors demonstrate increasing oscillatory behavior as the magnitude of λ_i increases (BrianDavies et al., 2001). The Graph Fourier Transform (GFT) of the vertex coordinates is defined as its projection onto the eigenvectors of the graph, i.e., $\bar{\mathbf{v}} = \mathbf{U}^T\mathbf{v}$ and the inverse GFT is given by $\mathbf{v} = \mathbf{U}\bar{\mathbf{v}}$.

4 BLOCK BASED SPECTRAL PROCESSING USING OI

As mentioned earlier, calculating the graph Laplacian eigenvalues of the mesh geometry can become restrictive as the density of the models increases. To overcome this limitation, several approaches suggest processing large meshes into parts (Cayre et al., 2003; Lalos et al., 2015). Thus, we assume the original 3D mesh is partitioned into k non-overlapping parts using the MeTiS algorithm described in (Karypis and Kumar, 1998). Processing of a single mesh in parts usually results in a loss of reconstruction quality that is attributed to the dislocation of the vertices that lie on the edges of each sub mesh. These phenomena, also known as edge effects, can be mitigated by processing overlapped submeshes (Cayre et al., 2003; Lalos et al., 2015). Therefore each submesh is extended with the neighbors of the boundary nodes of adjacent submeshes consisting in total of n_d nodes. This operation reduces the error introduced when increasing the number of sub meshes.

The evaluation of the eigenvectors of the respective matrix $\mathbf{L}[i]\ \forall i = 1, \ldots, k$ requires $O(kn_d^3)$ floating point operations. To minimize this complexity, we suggest exploiting the coherence between the spectral components of the different submeshes using OI (Golub and Van Loan, 2012). This assumption is strongly based on the observation that submeshes of the same mesh maintain similar geometric characteristics and connectivity information.

The Orthogonal Iteration is an iterative procedure, that can be used to compute the singular vectors corresponding to the dominant singular values of a symmetric, nonnegative definite matrix. Alternatively, to the OI one could use Lanczos approach. However, the initialization of OI to a starting subspace close to the subspace of interest leads to a very fast solution. This property is efficiently exploited when processing sequential submeshes, leading to a lower total complexity as compared to the complexity of the Lanczos approach. Building on this line of thought we suggest evaluating the c eigenvectors corresponding to the c lowest eigenvalue of $\mathbf{L}[i]$ each submesh i, $\mathbf{U}_c[i] = [\mathbf{u}_1, \ldots, \mathbf{u}_c] \in \Re^{n_d \times c}$ according to Algorithm 1, where $\mathbf{R}_i = (\mathbf{L}[i] + \delta\mathbf{I})^{-1}$ and δ is a small positive scalar value that ensures positive definiteness of $\mathbf{R}_i$. Matrix $\mathbf{I}$ is the identity matrix of size $n_d \times n_d$. At this point it should be noted that the projected coefficients $\mathbf{R}_i^z\mathbf{U}(t-1)$ are estimated very efficiently using sparse linear system solvers (Sorkine, 2005). Depending on the choice of power value z, we obtain alternative iterative algorithms with different convergence properties. The convergence rate of OI depends on $|\lambda_{c+1} / \lambda_c|^z$ where λ_{c+1} is the $(c+1)$-st largest eigenvalue of $\mathbf{R}_i$ (Zhang, 2009). To preserve orthonormality, it is important that the initial subspace $\widetilde{\mathbf{U}}_c[0]$ is orthonormal. For that reason, $\widetilde{\mathbf{U}}_c[0]$ is estimated by applying SVD directly on the first selected sub mesh[1], while the following subspaces $\widetilde{\mathbf{U}}_c[i]$, $i = 2, \ldots, k$ are adjusted using Algorithm 1. The initial submesh is

Algorithm 1: OI update process for each submesh i (**OI**).

```
U(0) ← U_c[i−1];
for t ← 1 to t_max do
    U(t) ← Onorm(R_i^z U(t−1));
end
U_c[i] ← U(t);
```

[1]Please note that the selection of the initial sub mesh does not affect the transient behavior of the algorithm

selected in a random order and the subsequent ones are processed in a topologically sorted order.

The orthonormalization of the estimated subspace can be performed using a number of different choices(Hua, 2004) that affect both complexity and performance.The most widely adopted are the Householder Reflections (HR), Gram-Schmidt (GS) and Modified GramSchmidt (MGS) methods. Although, the aforementioned variants exhibit different properties related to the numerical stability and computational complexity, the **Onorm**$(\cdot)$ step is performed using HR.

5 DYNAMIC OI FOR STABLE RECONSTRUCTION

In application scenarios where the original mesh is known, we propose a flexible solution that automatically identifies the optimal subspace size c that satisfies a specific reconstruction quality criterion. This novel extension can be used for improving the reconstruction quality in special cases where the coherence between submeshes is not strong enough e.g. different density, difference in geometry. The identification is performed sequentially, based on user defined thresholds, that determine the lower and higher "acceptable" quality of the reconstructed submeshes at the decoder side. In practical scenarios it is reasonable to assume that the feature vectors $\mathbf{E}[i] = \mathbf{U}_c^T[i]\mathbf{v}[i]$ of each block $\mathbf{v}[i]$ "live" in subspaces $\mathbf{U}_c[i]$ of different sizes. The subspace size c_i of the incoming data block $\mathbf{v}[i]$, should be carefully selected so that the relevant submesh vertices are identified with the minimum loss of information. To quantify this loss at each iteration t, we suggest using the l_2-norm of the following mean residual vector:

$$\mathbf{e}(t) = \sum_{j\in\{x,y,z\}} \left(\mathbf{v}_j[i] - \mathbf{U}_c[i]\,\mathbf{U}_c^T[i]\,\mathbf{v}_j[i]\right) \qquad (4)$$

where each $\mathbf{v}_j[i]$, $\forall\{x,y,z\}$ correspond to the $n_d \times 1$ vector with the x,y and z coordinates of the submesh i vertices. When the l_2 norm value of this metric is below a given threshold $\|\mathbf{e}(t)\|_2 < \varepsilon_h$ the loss of information during the spectral processing steps is not easily perceived. To reduce the residual error $\mathbf{e}(t)$, we suggest adding one normalized column in the estimated subspace $\mathbf{U}_c(t) = [\mathbf{U}_c(t-1)\ \ \mathbf{e}(t-1)/\|\mathbf{e}(t-1)\|_2]$ and then perform orthonormalization, e.g.,

$$\mathbf{U}_c(t) = Onorm\left\{\mathbf{R}^z[i][\mathbf{U}_c(t-1)\ \ \frac{\mathbf{e}(t-1)}{\|\mathbf{e}(t-1)\|_2}]\right\} \qquad (5)$$

Similarly, when the value of the reconstruction quality metric is less than a user determined lower bound ε_l the subspace size is decreased by 1 by simply selecting the first $c_i - 1$ columns of $\mathbf{U}_c(t)$. This procedure is repeated until the value of the metric lies within the range $(\varepsilon_l, \varepsilon_h)$, allowing the user to easily trade the reconstruction quality with the computational complexity. To summarize, Algorithm 2 presents the steps of the dynamic OI approach.

Algorithm 2: Dynamic OI for each submesh i (**DOI**).

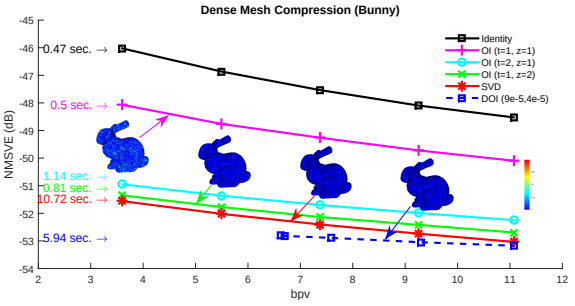
$\mathbf{U}(0) = \mathbf{U}_{i-1}$;
$c_i \leftarrow (i > 0)\ ?\ c_{i-1} : c$;
for $t = 1, 2, \ldots$ **do**
  $\mathbf{U}(t) = [u_1 \ldots u_{c_t}] = \mathbf{Onorm}(\mathbf{R}_i^z U(t-1))$;
  $\mathbf{e}(t) = \sum_{j\in\{x,y,z\}} \left(\mathbf{v}_{ij} - \mathbf{U}(t)\mathbf{U}(t)^T\mathbf{v}_{ij}\right)$;
  if $\|\mathbf{e}(t)\|_2 < \varepsilon_l$ **then**
    $\mathbf{U}(t) = [\mathbf{U}(t-1)\ \ \frac{\mathbf{e}(t)}{\|\mathbf{e}(t)\|_2}]$; $c_i \leftarrow c_i + 1$;
  else if $\|\mathbf{e}(t)\|_2 > \varepsilon_h$ **then**
    $\mathbf{U}(t) = [u_1 \ldots u_{c_t}]$; $c_i \leftarrow c_i - 1$;
  else
    break;
  end
end
$\mathbf{U}_i = \mathbf{U}(t)$;

6 APPLICATIONS

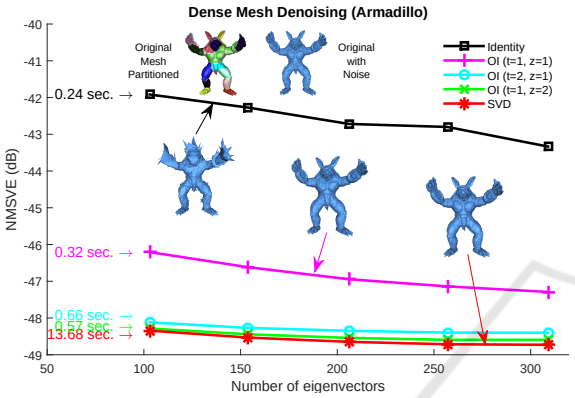
The primary purpose of this work is the creation of a framework for fast and effective spectral processing of large 3D meshes. In this section we present two case studies a) compression, b) denoising where the proposed schema can be applied.

6.1 Block based Spectral Compression of 3D Meshes

The spectral compression and reconstruction of static meshes utilize the subspace $\widetilde{\mathbf{U}}_c[i]$ for encoding and decoding the raw geometry data. During the encoding step, the dictionary $\widetilde{\mathbf{U}}_c[i]$ is evaluated, either by direct SVD or by executing a number of OI on $\mathbf{R}^z[i]$, and is used for providing a compact representation of the Euclidean coordinates of each submesh, e.g. for coordinates $\mathbf{v}_x[i] \in \Re^{n_d\times 1}$, $\mathbf{E}[i] = \mathbf{U}_\mathbf{c}^\mathbf{T}[i]\mathbf{v}[i]$, where $\mathbf{E}[i] \in \Re^{c\times 1}$ and $c << n_{d_i}$. At the decoder side the original 3D vertices of each submesh are reconstructed from the feature vector $\mathbf{E}[i]$ and the dictionary $\mathbf{U_c}[i]$ according to : $\tilde{\mathbf{v}}[i] = \mathbf{U}_c[i]\mathbf{E}[i]$. Note that the subspace size c remains fixed in the OI case, satisfying fast streaming scenarios, while DOI approach aims at providing high and stable reconstruction accuracy. It is important to mention that the only information transmitted from the sender is the connec-



(a)



(b)

Figure 1: (a) Compression Study: NMSVE vs bpv for the Bunny model (34,817 vertices) was partitioned into 70 blocks with about 512 vertices per block. (b) Coarse-to-fine Denoising study using the Armadillo model (20,002 vertices) was partitioned into 20 submeshes each comprising of around 990 vertices with zero-mean Gaussian noise $\mathcal{N}(0,0.2)$.

tivity of the mesh and the c respective spectral coefficients of each block. At the receiver's side, the dictionary $\widetilde{\mathbf{U}}_c[i]$ is evaluated utilizing the connectivity information. For the decoding process, the received spectral coefficients and the dictionary are used to retrieve the original Euclidean coordinates $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{i}, \hat{\mathbf{z}}_\mathbf{i}$, e.g. $\hat{\mathbf{x}}_\mathbf{i} = \widetilde{\mathbf{U}}_c[i]\mathbf{s}_{x_i}$. Spectral compression enables aggressive compression ratios, without introducing a significant loss on the visual quality (Karni and Gotsman, 2000).

6.2 Block based Spectral Denoising of 3D Meshes

Bilateral techniques have been used as mesh denoising method in many studies (Fleishman et al., 2003), (Zheng et al., 2011), (Zhang et al., 2015) by iteratively adjusting the face normals and vertices. In this section, we suggest executing a coarse-to-fine spectral denoising method that initially filters out the high spectral frequencies using the aforementioned

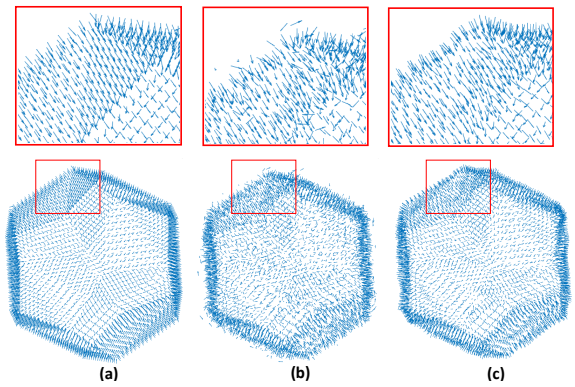


Figure 2: (a) Normals vector of original mesh, (b) Normals vector of noisy mesh, (c) Smoothed normals vector.

approach and then performs a fine denoising step using a two stage bilateral technique. The use of the coarse step significantly accelerates the convergence of the fine since it filters out the noise that appears in the higher frequencies, providing a set of normal vectors that are closer to the normal vectors of the original model, as it is clearly shown in the dodecahedron model Fig. 2.

We finally show that the fine technique can be also considered as Graph Spectral Processing approach. If we denote with $\hat{\mathbf{v}}[i] = \mathbf{U}_c\mathbf{U}_c^T\mathbf{v}[i]$ the vertices of the coarse denoised i submesh, then each face can be represented by its centroid point $\mathbf{m}_i$, and its outward unit normal:

$$\hat{\mathbf{n}}_{m_i} = \frac{(\hat{\mathbf{v}}_{i_2} - \hat{\mathbf{v}}_{i_1}) \times (\hat{\mathbf{v}}_{i_3} - \hat{\mathbf{v}}_{i_1})}{\left\|(\hat{\mathbf{v}}_{i_2} - \hat{\mathbf{v}}_{i_1}) \times (\hat{\mathbf{v}}_{i_3} - \hat{\mathbf{v}}_{i_1})\right\|}, \forall\, i = 1, n_f. \quad (6)$$

where $\hat{\mathbf{v}}_{i_1}, \hat{\mathbf{v}}_{i_2}, \hat{\mathbf{v}}_{i_3}$ are the vertices that are related with face f_i and $\hat{\mathbf{n}}_m = [\hat{\mathbf{n}}_{m_1}^T\ \hat{\mathbf{n}}_{m_2}^T \cdots \hat{\mathbf{n}}_{n_f}^T] \in \Re^{3n_f \times 1}$.

The bilateral technique estimates the new face normals $\mathbf{n}_i$ using a normal guidance unit vector $\mathbf{g}_i$, which it is calculated as a weighted average of normals in a neighborhood of i is computed by:

$$\hat{\mathbf{n}}_{m_i} = \frac{1}{W_i} \sum_{f_j \in \mathcal{N}_{f_i}} A_j K_s(\mathbf{m}_i, \mathbf{m}_j) K_r\left(\mathbf{n}_{m_i}, \mathbf{n}_{m_j}\right) \mathbf{n}_{m_j} \quad (7)$$

where $\mathcal{N}_{f_i}$ is the set of faces in a neighborhood of f_i, A_j is the area of face f_j, W_i is a weight that ensures that $\hat{\mathbf{n}}_{m_i}$ is a unit vector and K_s, K_r are the spatial and range Gaussian kernels. More specifically, K_s is monotonically decreasing with respect to the distance of the centroids $\mathbf{m}_i$ and $\mathbf{m}_j$ which lie on the mesh surface, while K_r is monotonically decreasing with the proximity of the guidance normals that lie on the unit sphere:

$$K_s(\mathbf{m}_i,\mathbf{m}_j) = exp\left(-\frac{\left\|\mathbf{m}_i-\mathbf{m}_j\right\|^2}{2\sigma_s^2}\right), \quad (8)$$

$$K_r\left(\mathbf{n}_{m_i},\mathbf{n}_{m_j}\right) = exp\left(-\frac{\left\|\mathbf{n}_{m_i}-\mathbf{n}_{m_j}\right\|^2}{2\sigma_r^2}\right) \quad (9)$$

The Bilateral filter output is then used to update the vertex positions in order to match the new normal directions $\mathbf{n}_{m_i}$, according to the iterative scheme proposed in (Zhang et al., 2015). More specifically, the vertex positions $\hat{\mathbf{v}}_{i_1}$, $\hat{\mathbf{v}}_{i_2}$, $\hat{\mathbf{v}}_{i_3}$ of a face f_i are updated in an iterative manner, according to:

$$\hat{\mathbf{v}}_{i_j}^{(t+1)} = \hat{\mathbf{v}}_{i_j}^{(t)} + \frac{1}{\left|\mathcal{F}_{i_j}\right|}\sum_{z\in\mathcal{F}_{i_j}}\hat{\mathbf{n}}_{m_z}\left[\hat{\mathbf{n}}_{m_z}^T\left(\mathbf{m}_i^{(t)}-\hat{\mathbf{v}}_{i_j}^{(t)}\right)\right] \quad (10)$$

$$\mathbf{m}_i^{(t)} = \left(\hat{\mathbf{v}}_{i_1}^{(t)}+\hat{\mathbf{v}}_{i_2}^{(t)}+\hat{\mathbf{v}}_{i_3}^{(t)}\right)/3 \quad (11)$$

where (t) denotes the iteration number, $\mathcal{F}_{i_j}$ is the index set of incident faces for $\hat{\mathbf{v}}_{i_j}$. This iterative process can be considered as a gradient descent process that is executed for minimizing the following energy term across all faces

$$\sum_{z\in\mathcal{F}_{i_j}}\left|\hat{\mathbf{n}}_{m_z}^T\left(\mathbf{m}_i^{(t)}-\hat{\mathbf{v}}_{i_j}^{(t)}\right)\right|^2,\ j=1,2,3. \quad (12)$$

This term penalizes displacement perpendicular to the tangent plane defined by the vertex position $\hat{\mathbf{v}}_{i_j}^{(t)}$ and the local surface normal $\hat{\mathbf{n}}_{m_z}$

Bilateral filter as a graph based transform:

Consider an undirected graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ where the nodes $\mathcal{V}=\{1,2,\ldots,n\}$ are the normals $\mathbf{n}_{m_i}$, associated with the centroids $\mathbf{m}_i$ and the edges $\mathcal{E}=\left\{(i,j,c_{ij})\right\}$ capture the similarity between two normals as given by the bilateral weights in eq. (8), (9). The input normals can be considered as a signal defined on this graph $\mathbf{n}_i:\mathcal{V}\rightarrow\Re^{3\times1}$ where the signal value at each node correspond to the normal vector. Let $\mathbf{C}$ be the adjacency matrix with the bilateral weights and $\mathbf{D}=diag\left\{W_1,\ldots,W_{n_f}\right\}$ the diagonal degree matrix, then eq. (7) can be written as:

$$\begin{aligned}\hat{\mathbf{n}} &= \mathbf{D}^{-1}\mathbf{C}\mathbf{n}\\ &= \mathbf{D}^{-1/2}\mathbf{D}^{-1/2}\mathbf{C}\mathbf{D}^{-1/2}\mathbf{D}^{1/2}\mathbf{n}\\ \mathbf{D}^{1/2}\hat{\mathbf{n}} &= (\mathbf{I}-\mathbf{L})\mathbf{D}^{1/2}\mathbf{n}\\ \mathbf{D}^{1/2}\hat{\mathbf{n}} &= \underbrace{\mathbf{U}}_{IGFT}\ \underbrace{(\mathbf{I}-\Lambda)}_{\substack{Spectral\\ response}}\ \underbrace{\mathbf{U}^T}_{GFT}\mathbf{D}^{1/2}\mathbf{n}. \quad (13)\end{aligned}$$

Thus, it is clearly shown that the Bilateral filter can be considered as a frequency selective graph transform with a spectral response that corresponds to a linear decaying function, meaning that it tries to preserve the low frequency components and attenuate the high frequency ones.

7 PERFORMANCE EVALUATION

In the following section, we evaluate the presented framework in two different case studies: i) block based mesh compression and ii) block based mesh denoising, that effectively take advantage of the spectral coherence between different blocks utilizing OI. Each case study is examined using a static large ($\geq 2\cdot10^4$ vertices) and very large ($\geq 1\cdot10^6$ vertices) mesh partitioned using MeTis. All simulations were performed on an Intel Core i7-4790 (3.6 GHz) processor with 8GB RAM. The compression efficiency of the geometry is measured in bits-per-vertex ($bpv = 3\cdot q_c\cdot c\cdot k/n_d$) where q_c are the bits used for uniformly quantizing the feature vectors (q_c = 12 bits) and c the total components kept from each submesh. This metric encapsulates the feature vectors for each processed block, ignoring the mesh connectivity which can be effectively compressed through any state-of-the-art connectivity encoder (Rossignac, 1999). To evaluate the reconstruction quality of our proposed method, it is necessary to capture the distortion between the original and the approximated frame. For this task, we chose the normalized mean square visual error (NMSVE) (Karni and Gotsman, 2000) calculated as:

$$\frac{1}{2n}\left(\|\mathbf{v}-\tilde{\mathbf{v}}\|_{l_2}+\|GL(\mathbf{v})-GL(\tilde{\mathbf{v}})\|_{l_2}\right) \quad (14)$$

where $GL(\mathbf{v}_i)=v_i-(\sum_{j\in N(i)}d_{ij}^{-1}v_j)/(\sum_{j\in N(i)}d_{ij}^{-1})$, $\mathbf{v},\tilde{\mathbf{v}}\in\mathbb{R}^{3n\times1}$ represent vectors that contain the original and reconstructed vertices respectively, and d_{ij} denotes the Euclidean distance between i and j.

7.1 Compression Results

The NMSVE vs bpv results are shown in Fig. 1 (a) for the Bunny model. Note that the execution times shown next to each line encapsulate the respective time needed to construct $\mathbf{R}^z, z\geq1$, and to run the respective number of OI. By inspecting the figure, it can be easily concluded that the quality of the OI method performs almost the same as with SVD, especially when the number of iterations increases. At this point it should be noted that the benefits of our method are directly related to the size of each block. The theoretical complexities of the proposed schemes are in tandem with the measured times. More specifically,

the OI approach for the Bunny mesh can be executed up to 20 times faster than the direct SVD approach. Although running more OI iterations yields a better NMSVE, converging towards the (optimal) SVD result, it comes at the cost of a linear increase in the decoding time. On the other hand, one iteration of $\mathbf{R}^2$ achieves lower visual error as executing two OI, in considerably less time. Moreover, DOI provides a stable reconstruction accuracy (see Fig. 3 showing the per submesh error) that can easily be adjusted by the defined thresholds. By inspecting also Fig. 3, it is obvious that there is a coherence between submeshes since there are very few abrupt changes in the 'ideal' value of subspace size that is required to satisfy a predefined reconstruction quality.

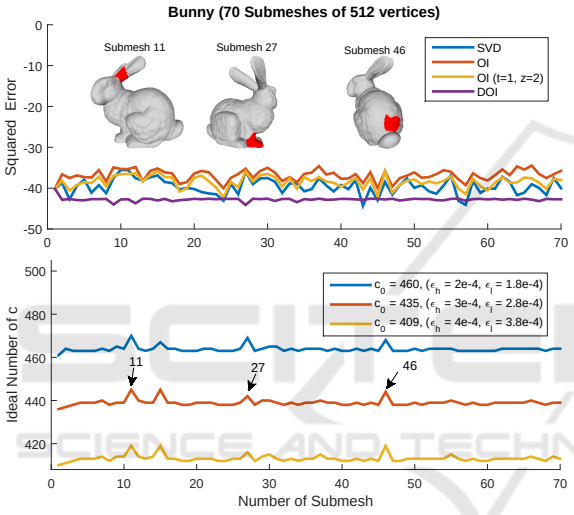


Figure 3: [Up] Squared Error per submesh for different approaches. [Bottom] Ideal value c of subspace size per submesh.

However, this comes with a slight increase on the execution time (more OI) as well as a significant increase on the final compression rate (bpv) captured in Fig. 1 (a) as a right shifting of the plot. The shifting is more obvious when the initial value of c is small (more OI iterations are necessary for achieving the accuracy threshold).

7.2 Denoising Results

Similar conclusions are also drawn in a coarse-to-fine denoising setup where OI method are used as a preprocessing, "smoothing" step, before applying a conventional spectral bilateral filtering Fig. 1 (b). Fig. 4 shows the effects of the coarse denoising step in the execution time of the guided mesh approach (Zhang et al., 2015). By inspecting the number of iterations required for achieving a given reconstruction quality,

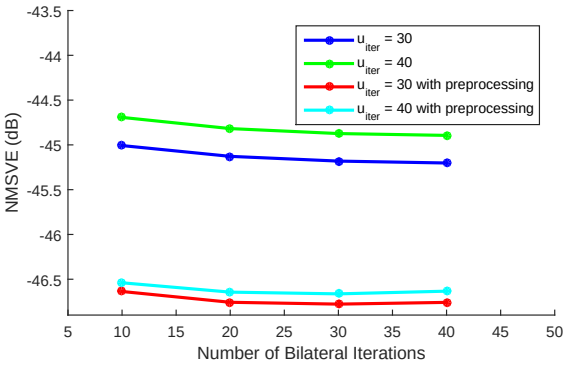


Figure 4: Coarse denoising step increases the reconstruction quality. In other words, less faces/vertices updates are required in order to achieve the same quality.

Table 1: Execution time and face angle difference θ for different cases of $\mathbf{R}^z$ and SVD.

	Twelve		Fandisk	
	t	θ	t	θ
$\mathbf{R}^1$	0.031	11.57	0.077	16.36
$\mathbf{R}^2$	0.049	10.26	0.110	14.75
$\mathbf{R}^3$	0.099	13.97	0.170	14.54
$\mathbf{R}^4$	0.114	13.84	0.202	14.54
$\mathbf{R}^5$	0.136	13.7	0.242	14.44
$\mathbf{R}^6$	0.142	13.59	0.297	14.5
$\mathbf{R}^7$	0.157	13.57	0.313	14.52
$\mathbf{R}^8$	0.184	13.55	0.355	14.84
$\mathbf{R}^9$	0.201	13.53	0.407	15.6
SVD	0.901	9.83	1.953	14.56

it can be easily noted that the coarse denoising step accelerates the convergence rate of the fine denoising approach, thus significantly improving the total execution time of the whole denoising process.

In Figs. 5 and 6, it can be easily observed that the presented OI method can be employed by any other state-of-the-art (SoA) denoising method as a preprocessing step (Zheng et al., 2011; He and Schaefer, 2013; Zhang et al., 2015), optimizing both its reconstruction quality and its computational complexity. The use of the coarse step significantly accelerates the convergence of the fine reducing the face/vertex update iterations required for achieving a specific reconstruction quality.The reconstruction benefits can be easily identified by inspecting Fig. 5 which presents denoising results of SoA methods (first row) and the corresponding results after using the OI approach as a preprocessing step (second row).

Moreover, we also examined different combinations of z (power of $\mathbf{R}$) and number of iterations in the denoising setup. Figure 7 shows the results of the coarse denoising step using OI for different values of z. Higher values, result in higher accuracies as

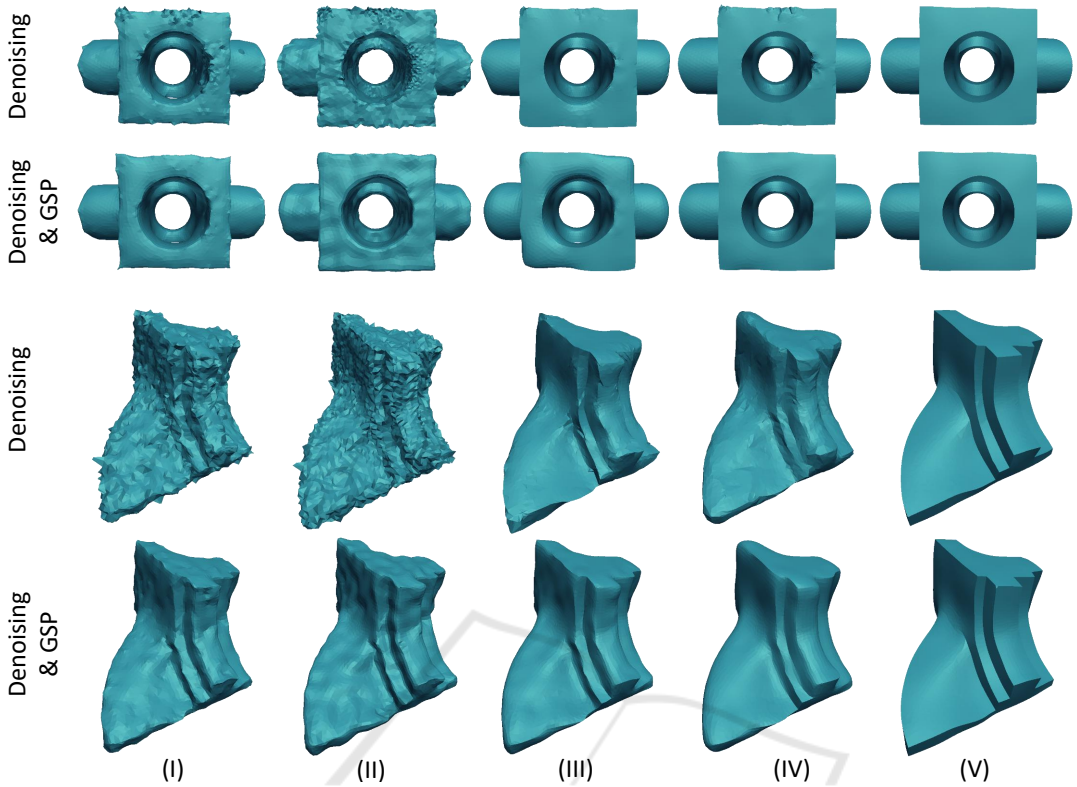


Figure 5: Coarse denoising improves the efficiency of the following SoA approaches: (i) bilateral (Fleishman et al., 2003), (ii) non iterative (Jones et al., 2003), (iii) fast and effective (Sun et al., 2007), (iv) bilateral normal [24], (v) guided normal filtering (Zhang et al., 2015) (zero-mean Gaussian noise $\mathcal{N}(0, 0.7)$).

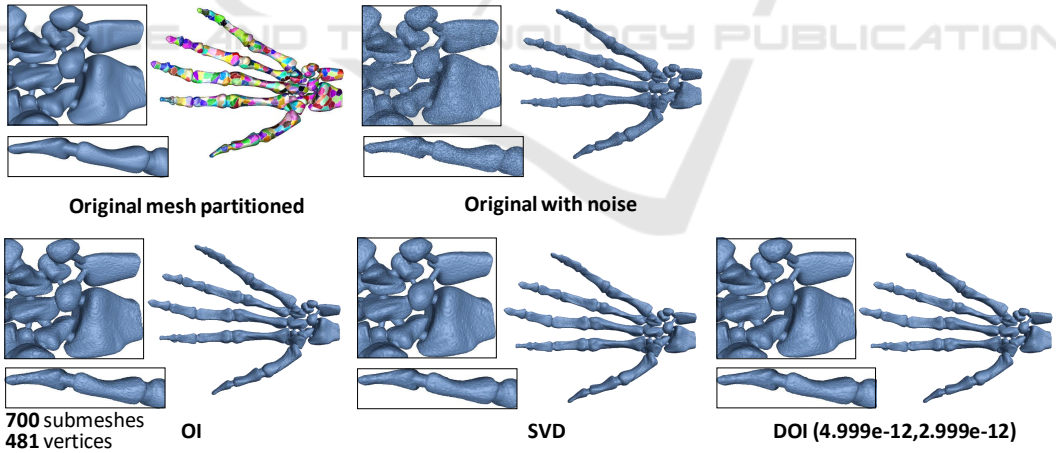


Figure 6: Graph Spectral processing of Hand (327,323 vertices) using $c = 47$ eigenvectors, by applying the (a) OI method, (b) traditional SVD method, (c) Dynamic OI method.

compared to the direct application of SVD. While it should be noted that for $z > 4$ the results are identical identical with that achieved by applying the direct SVD.

We evaluated the effects of executing several OI either on $\mathbf{R}^z$ or on $\mathbf{R}$ in CAD and scanned 3D models, using the NMSVE and the angle difference between the normal of the ground truth face and the corresponding normal of the reconstructed face, averaged over all faces (θ). The differences between the SVD and OI, in terms of both reconstruction quality and execution time, are presented in Tables 1 and 2. It is clearly shown that the application of OI on $\mathbf{R}^z$ results in faster execution times than the applica-

Table 2: NMSVE and Execution Times.

		NMSVE (dB)		Time (sec.)		
	ver/clust	**SVD**	**OI**	**SVD**	**OI**	**Speed-up**
Armadillo	10%	-47.5668	-47.4145	8.65	0.40	22x
	15%	-47.6641	-47.5263	8.80	0.53	16x
	20%	-47.7428	-47.6195	9.06	0.65	14x
	25%	-47.8301	-47.7097	9.16	0.80	12x
Hand	10%	-59.0354	-58.8521	48.36	0.54	89x
	15%	-58.872	-58.7107	48.85	0.69	70x
	20%	-58.6218	-58.489	49.14	1.05	46x
	25%	-58.3314	-58.2215	49.53	1.31	37x

Table 3: Time performance for different models. k_{iter} presents the number of iterations for bilateral filtering normal executed in $t_{k_{iter}}$, u_{iter} is the number of iterations for vertex update executed in $t_{u_{iter}}$ and N is the number of points searched for finding ideal patches (according to (Zhang et al., 2015)) executed in t_N. t_{cd} corresponds to execution time for coarse denoising, features extraction and level noise estimation, t_{cd} represents the execution time for fine denoising and t_{tot} is the total time expressed in seconds.

	Guided normal filtering (GNF) (Zhang et al., 2015)						**GNF using Coarse denoising step**					
	$\mathbf{k_{iter}}$	$\mathbf{t_{k_{iter}}}$	$\mathbf{t_{u_{iter}}}$	**N**	$\mathbf{t_N}$	t_{tot}	$\mathbf{t_{cd}}$	$\mathbf{t_{fn}}$	$\mathbf{t_{u_{iter}}}$	**N**	$\mathbf{t_N}$	$\mathbf{t_{tot}}$
Block	40	200.31	21.88	17550	10.63	232.82	7.08	32.59	14.78	4457	2.68	**57.13** -75%
Twelve	75	273.07	7.64	9216	8.11	289.55	2.25	18.67	11.54	2297	2.31	**34.77** -88%
Sphere	30	106.91	17.60	20882	6.53	131.05	8.19	18.10	10.44	3635	1.22	**37.95** -71%
Fandisk	50	257.13	10.61	12946	12.19	279.94	7.53	44.82	10.82	4464	4.34	**67.51** -75%

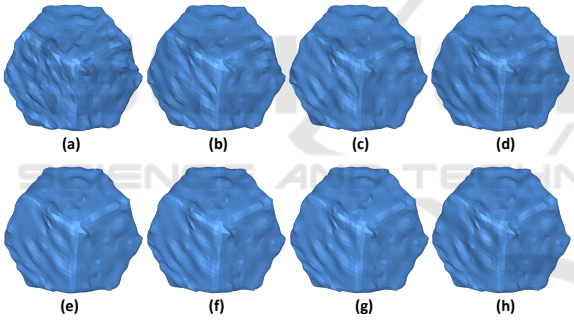


Figure 7: Coarse denoising results for different cases of $\mathbf{R}^z$, (a) $z = 1$ (b) $z = 2$ (c) $z = 3$ (d) $z = 4$ (e) $z = 5$ (f) $z = 6$ (g) $z = 7$ (h) SVD.

tion on $\mathbf{R}$. This result is attributed to the facts that: i) the execution of OI on $\mathbf{R}$ requires z times more iterations to converge than the execution on $\mathbf{R}^z$ ii) while the evaluation of a matrix-matrix product (e.g., $\mathbf{R} \times \mathbf{R}$, e.t.c.) is much less computationally demanding than the orthonormalization step. These effects become more apparent in dense models (see. Table 2). Finally, in Table 3 we present the execution times of the proposed denoising approach as compared to the Guided Mesh algorithm (Zhang et al., 2015), which provided the best results among the SoA competitors. Note that while both approaches, are based on a sequential update of the face normals and vertices, the GNF with coarse denoising, results to lower execution times (12x-88x). This reduction is attributed to the application of the coarse denoising step that filters out the high frequency components, accelerating the convergence speed of the necessary corrections/ adjustments of the vertex positions.

8 DISCUSSION AND CONCLUSION

In this paper, we introduced a fast and efficient way of performing spectral processing of 3D meshes ideally suited for real time applications. The proposed approach apply the problem of tracking graph Laplacian eigenspaces via orthogonal iterations, exploiting potential spectral coherences between adjacent parts. The thorough experimental study on a vast collection of 3D meshes that represent a wide range of CAD and scanned Models showed that the subspace tracking approaches allow the robust estimation of dictionaries at significantly lower execution times compared to the direct SVD implementations. Despite the superiority of OI based approaches when compared to the direct SVD, the optimal subspace size should be carefully selected in order to simultaneously achieve the highest reconstruction quality and fastest compression times. One interesting future direction, is to propose an automatic modeled based approach for identifying the subspace size where the features lie together with the level of noise. In addition, the approximation artifacts that occur in a single block may

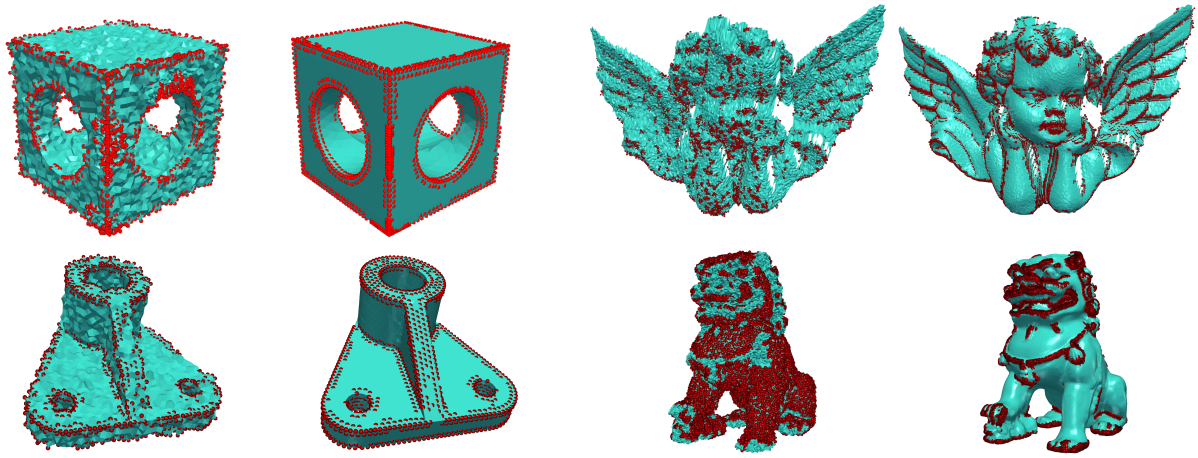


Figure 8: Identification of sharp and small scale geometric features in noisy objects with and without DAOI step.

slightly increased and propagated when moving to the subsequent ones. To address the latter issue, we suggest to either re-initialize the subspace of interest using the SVD or execute an DAOI update. At this point we would like also to highlight another strong point of the proposed GSP scheme. The proposed GSP approach increases significantly the accuracy of state of the art schemes, focusing on identifying sharp and small scale geometric features. In Fig. 8, we present the identified features in the case that we use as input a noisy model or the same noisy model after applying the GSP approach. It is clearly shown that the GSP approach increases significantly the robustness of the feature identification approach, allowing the accurate detection of high frequency featurea that could be further used by several feature aware compression, completion and denoising approaches. Without any doubt, the presented signal processing approaches are expected to provide a novel insight at key areas, e.g., compression and feature preserving denoising where high potential for novel improvements is feasible in the near future.

REFERENCES

Alexiadis, D. S., Zarpalas, D., and Daras, P. (2013). Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339–358.

BrianDavies, E., Gladwell, G. L., Leydold, J., and Stadler, P. F. (2001). Discrete nodal domain theorems. *Linear Algebra and its Applications*, 336(1-3):51–60.

Cai, W., Leung, V. C. M., and Chen, M. (2013). Next Generation Mobile Cloud Gaming. *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pages 551–560.

Cayre, F., Rondao-Alface, P., Schmitt, F., Macq, B., and Maıtre, H. (2003). Application of spectral decomposition to compression and watermarking of 3d triangle mesh geometry. *Signal Processing: Image Communication*, 18(4):309–319.

Comon, P. and Golub, G. H. (1990). Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE*, 78(8):1327–1343.

Fleishman, S., Drori, I., and Cohen-Or, D. (2003). Bilateral mesh denoising. *ACM Trans. Graph.*, 22(3):950–953.

Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.

Gotsman, C. (2003). On graph partitioning, spectral analysis, and digital mesh processing. In *Shape Modeling International, 2003*, pages 165–171. IEEE.

He, L. and Schaefer, S. (2013). Mesh denoising via l0 minimization. *ACM Trans. Graph*, 32(4):64:164:8.

Hua, Y. (2004). Asymptotical orthonormalization of subspace matrices without square root. *IEEE Signal Processing Magazine*, 21(4):56–61.

Jones, T. R., Durand, F., and Desbrum, M. (2003). Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph*, 22(3):943949.

Karni, Z. and Gotsman, C. (2000). Spectral compression of mesh geometry. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 279–286.

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392.

Kim, B. and Rossignac, J. (2005). Geofilter: Geometric selection of mesh filter parameters. In *Computer Graphics Forum*, volume 24, pages 295–302. Wiley Online Library.

Lalos, A. S., Nikolas, I., and Moustakas, K. (2015). Sparse coding of dense 3d meshes in mobile cloud applications. In *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 403–408. IEEE.

Lalos, A. S., Nikolas, I., Vlachos, E., and Moustakas, K. (2017). Compressed sensing for efficient encoding of dense 3d meshes using model-based bayesian learning. *IEEE Transactions on Multimedia*, 19(1):41–53.

Lévy, B. (2006). Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry. In *Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on*, pages 13–13. IEEE.

Mekuria, R., Sanna, M., Izquierdo, E., Bulterman, D. C. A., and Cesar, P. (2014). Enabling geometry-based 3-D tele-immersion with fast mesh compression and linear rateless coding. *IEEE Transactions on Multimedia*, 16(7):1809–1820.

Ohbuchi, R., Takahashi, S., Miyazawa, T., and Mukaiyama, A. (2001). Watermarking 3d polygonal meshes in the mesh spectral domain. In *Graphics interface*, volume 2001, pages 9–17.

Rossignac, J. (1999). Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics*, 5(1):47–61.

Saad, Y. (2016). Analysis of subspace iteration for eigenvalue problems with evolving matrices. *SIAM Journal on Matrix Analysis and Applications*, 37(1):103–122.

Sorkine, O. (2005). Laplacian Mesh Processing. In Chrysanthou, Y. and Magnor, M., editors, *Eurographics 2005 - State of the Art Reports*. The Eurographics Association.

Sun, X., Rosin, P. L., Martin, R. R., and Langbein, F. C. (2007). Fast and effective feature-preserving mesh denoising. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 13(5):925938.

Taubin, G. (1995). A signal processing approach to fair surface design. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 351–358, New York, NY, USA. ACM.

Taubin, G. (2000). Geometric signal processing on polygonal meshes.

Zhang, H., Van Kaick, O., and Dyer, R. (2010). Spectral mesh processing. In *Computer graphics forum*, volume 29, pages 1865–1894. Wiley Online Library.

Zhang, P. (2009). Iterative methods for computing eigenvalues and exponentials of large matrices.

Zhang, W., Deng, B., Zhang, J., Bouaziz, S., and Liu, L. (2015). Guided mesh normal filtering. *Pacific Graphics*, 34(7).

Zheng, Y., Fu, H., Au, O. K.-C., and Tai, C.-L. (2011). Bilateral normal filtering for mesh denoising. *IEEE Trans. Vis. Comput. Graph.*, 17(10):1521–1530.