Graph-Based Wavelet Multiresolution Modeling of Multivariate Terrain Data

Teodor Cioacă, Bogdan Dumitrescu and Mihai-Sorin Stupariu

Abstract Terrain data pose modeling challenges due to their high inherent redundancy and difficulty of identifying features at different levels of detail. We propose a multiresolution analysis framework based on a graph-based wavelet construction. Our approach produces a sequence of intermediate resolution approximations of the terrain model. The details pertaining to each resolution reveal scale-specific features. Using a guiding heuristic, the proposed wavelet construction also conserves salient features. Furthermore, the proposed framework allows both geometric and attribute vertex information and can be used for modeling tasks sharing the same characteristics and constraints with terrain modeling. In particular, our graph-based wavelet framework is an option for multiresolution filtering and feature classification or clustering.

1 Introduction

The particular problematics of terrain modeling span a broad range of quantitative and qualitative analysis tasks. Such tasks include, but are not limited to geometric representation, geomorphological feature identification and analysis, vegetation coverage analysis and extraction. The continuous advancements of 3D sensor scanning technologies and the increasing popularity of civil aerial drones further facilitate gathering high density point cloud terrain representations.

In this chapter, we describe an approach for multiresolution analysis of multivariate terrain data, which is also suitable for other data organized in

T. Cioacă (✉) · B. Dumitrescu
University Politehnica of Bucharest, Bucharest, Romania
e-mail: t.cioaca@gmail.com

B. Dumitrescu
e-mail: bogdan.dumitrescu@acse.pub.ro

M.-S. Stupariu · T. Cioacă
University of Bucharest, Bucharest, Romania
e-mail: stupariu@fmi.unibuc.ro

L. Stanković and E. Sejdić (eds.), *Vertex-Frequency Analysis of Graph Signals*, Signals and Communication Technology,
https://doi.org/10.1007/978-3-030-03574-7_15

similar structures. In general, the framework we propose can be easily adapted for performing multiresolution analysis on triangulated point clouds, meshes or graphs. Any of these data structures may describe an irregular domain where a multivariate signal is discretized. The samples are then concentrated in the vertices of these structures.

Given these aspects, the approach we will further describe capitalizes on recently developed methods for graph signal processing. More specifically, we explore a wavelet-based graph multiresolution construction that has a reduced approximation error and is computationally efficient. The graph wavelet interpretation facilitates tasks involving *vertex frequency analysis* or *multiresolution filtering and clustering*.

1.1 Terrain Data Representation

Typically, the geometric information describing terrain fragments is encoded in a digital elevation model (DEM) [36]. A triangulated irregular network (TIN) is a further specialization of a DEM where both point and primitive (triangle) sampling are allowed to be irregular, reducing the highly redundant information in flat regions. De Floriani and Magillo [13] provide the following definition:

> A terrain can be mathematically modeled as a function $z = f(x, y)$ mapping a point (x, y) in a domain D in the plane to its elevation value $f(x, y)$. In practice, the value of function f is known on a finite set S of points within D. A DEM provides an estimated value for function f at any point (x, y) of the domain, based on the values at the points of S. A DEM consists of a subdivision of the domain into cells and of a piece-wise interpolating function defined on such cells. A TIN is a DEM in which the domain subdivision is a triangular mesh, i.e., a set T of triangles such that:
>
> 1. the set of vertices of T is S
> 2. the interiors of any two triangles of T do not intersect
> 3. if the boundaries of two triangles intersect, then the intersection is either a common vertex, or a common edge.

We can easily conclude that the TIN definition is practically identical to that of a triangular mesh as generally understood in the field of Computer Graphics.

1.2 Challenges of Terrain Modeling From Raw Point Data

LiDAR data are usually dense and the information richness can pose a problem on its own since there is no intrinsic manner of reducing geometric or other attribute redundancy. However, model simplification is a well studied problem for which established solutions have been proposed in the field of Computer Graphics. Among the computationally efficient approaches, we mention *incremental thinning*, where geometric primitives are simply removed by following a local quality-based simplification criterion. For TIN models, Demaret et al. [14] have proposed an efficient coarsification algorithm via incremental point removal.

Suarez and Plaza [34] have introduced a coarsening algorithm applicable to right-triangulated irregular networks. Their method is especially efficient given that both the underlying geometry and triangle primitive connectivity is easier to control than in the case of general irregular networks as each vertex has a maximum of 8 directly connected neighbors. The actual coarsening is then achieved through edge midpoint split operations, which are also invertible.

Another problem posed by raw point clouds is that there is no immediate and accurate information concerning the ground or vegetation class of constituent points. While ground return information can help define a subset of soil-level points, the overall point cloud usually contains more ground-level information, as well as potentially unclassified data points. Evans and Hudak [15] have designed a *multiscale curvature classification algorithm* in order to filter and classify points present in raw LiDAR collections.

Silva et al. [2] examine four different algorithms for classifying ground from hovering points in point cloud terrain data sets. Among the algorithms discussed in their work, the progressive triangulated irregular network, implemented in the LAStools package [20], was proven to possess good performance characteristics (low RMSE and increased ground classification accuracy with respect to the ground truth).

1.3 Existing Solutions

Given that terrain modeling tasks pose similar problems to those encountered in point cloud, mesh or graph data processing, we will briefly review some of the milestone achievements from these areas.

Incremental Simplification of Point Clouds and Meshes

Historically, solutions to feature redundancy have been addressed separately from multiresolution analysis. The most common solution to this problem is incremental simplification. Modeling of terrain data is also possible with either point cloud or mesh simplification methods.

For point clouds, notable contributions are given in the work of Pauly et al. [29] where three simplification method categories are detailed: clustering, iterative simplification and particle simulation.

In the case of triangular or polygonal meshes, the additional information also facilitates the creation of reversible, continuous level-of-detail representations and mesh morphing. Schroeder et al. [32] were among the first authors to discuss triangular mesh simplification in a step-by-step manner known as *decimation*. Almost all decimation-based methods require the existence of a decimation criterion and can be summarized in two steps that are repeated until the target density is reached: *(i)* evaluate decimation criterion (usually a cost function estimating the impact of removing a set of primitives) and *(ii)* remove a primitive and re-triangulate the local geometry.

To address changes in the mesh connectivity resulting from primitive removal, Ronfard et al. [31] proposed a reversible operation called *edge collapse*. One of the most effective heuristic criteria for guiding these operations is the *quadric error metric (QEM)* formulation of Garland and Heckbert [17].

Wavelet Multiresolution Methods

Successful initiatives for adapting established Signal Processing methods for point cloud, mesh and graph analysis have been recorded.

To briefly summarizes notable adaptations of wavelet-like transforms on point clouds we mention the following works. Chen et al. [5] have introduced an SVD-based construction of so-called geometric wavelet point cloud analysis. Earlier, Choi et al. [6] applied a straightforward implementation of Swelden's *lifting scheme* [35] for denoising point sampled manifolds of low intrinsic dimension.

Among the first wavelet mesh multiresolution analysis methods is that of Lounsbery [26]. Using a subdivision operator on triangular meshes, Lounsbery introduced an analog of the refinement equation for functions defined on mesh structures instead on the real line. With the introduction of Swelden's *lifting scheme* [35], other approaches using this technique were developed. One example is the critically sampled design for irregular meshes described by Bertram [3]. His method is similar to that of Guskov et al. [18], where a geometry-based subdivision operator defines wavelet synthesis.

After mesh wavelet constructions, algorithms specifically design for more generic graph data were also introduced. We mention the method of Coifman et al. [12], which uses a diffusion operator to build a wavelet transform in the graph frequency domain. Hammond et al. [19] further proposed a method that, instead of diagonalizing the graph Laplacian matrix, a costly process for large graphs, uses approximations based on Chebyshev polynomials. In the graph spatial domain we mention Jansen's approach [21, 22], also adopted and extended by Wagner et al. [37], and by Martinez and Ortega [27].

Wavelet Multiresolution for Terrain Modeling

Among the first solutions specifically designed with terrain modeling in mind is the *wavelet triangular irregular networks* framework of Wu and Amaratunga [24]. This method is essentially a *lifting scheme* adaptation using an inverse subdivision prediction filter and resampling in the xy plane. The importance of wavelet coefficients, byproducts of multiresolution analysis, was emphasized by Beyer [4]. The author demonstrated that both terrain gradient and curvature information can be derived from these wavelet detail vectors. Kalbermatten's et al. [25] work further explores applications of wavelet transform in identifying multiscale geomorphological and geological features.

2 Terrain Modeling Using Graph-Based Wavelet Multiresolution Analysis

We now address the question of how a wavelet multiresolution framework can be defined for terrain models. From a signal processing perspective, DEMs are similar to images, which are simply two-dimensional signals defined over regular domains. When modeling data from LiDAR sources, TIN structures are better options because they allow for adaptive sampling in areas with high geometric redundancy. Building wavelets on these irregular domains would enable the analysis of terrain characteristics at different levels of resolution, which, in turn, can be perceived as information pertaining to a certain frequency band.

Our goal is threefold. First, we aim to expand previous methods and achieve multiresolution representation of multivariate terrain sets (having both geometric and attribute coordinates for each point). Second, by adopting a spatial domain graph lifting scheme construction, we maintain a balance between computational complexity and the range of potential practical applications of the method. Third, we wish to explore the semantic interpretation of detail vectors, leading to a correlation between vertices and levels of resolution. This interpretation would allow our method to be applicable to *vertex frequency analysis* tasks.

Both the construction and results presented in the following sections of this chapter have been partially described in our previous works [7, 9].

2.1 Classic Wavelet Analysis and Synthesis

Before detailing our wavelet multiresolution design for multivariate signals, we review the fundamental concepts in the one-dimensional case.

Problem Formulation

In general, wavelets are the building blocks of hierarchical function space approximations, i.e. $L^2(\mathbb{R}) \succ \ldots \succ V_n \succ V_{n-1} \succ \ldots \succ V_1 \succ V_0 \succ \ldots \succ \{0\}$, where V_j is a function space approximation at level j. We are interested in examining the connections between two consecutive approximations, V_{j+1} and V_j. Let $\Phi_j = \left[\ldots\ \phi_{j,k}\ \ldots\right]$ be the row vector of basis functions that generate V_j. Then a function $f_{j+1} \in V_{j+1}$ cannot be represented by using only the basis functions in V_j, but it can be expressed as a combination of the basis functions of the V_{j+1} space, that is

$$f_{j+1} = \Phi_{j+1}\mathbf{s}_{j+1}, \qquad (1)$$

where $\mathbf{s}_{j+1}$ is an infinite column vector of *scaling coefficients*. The complementary basis or *wavelet functions*, $\psi_{j,k}$, that generate the orthogonal subspace, correspond to the lost details W_j. Thus, the *direct sum decomposition* of the higher-resolution approximation, $V_{j+1} = V_j \oplus W_j$, can be expressed in terms of basis and wavelet functions as

$$f_{j+1} = \Phi_{j+1}\mathbf{s}_{j+1} = \Phi_j \mathbf{s}_j + \Psi_j \mathbf{w}_j, \tag{2}$$

where $\Psi_j = \left[\ldots \psi_{j,k} \ldots\right]$ and $\mathbf{w}_j$ is the infinite column vector of *wavelet coefficients*.

Both the scaling and wavelet coefficients can be written in terms of internal products between f_{j+1} and the dual basis functions of their corresponding spaces, i.e.:

$$\mathbf{s}_{j+1,k} = \left\langle \tilde{\phi}_{j+1,k}, f_{j+1} \right\rangle, \tag{3}$$

$$\mathbf{s}_{j,k} = \left\langle \tilde{\phi}_{j,k}, f_{j+1} \right\rangle, \tag{4}$$

$$\mathbf{w}_{j,k} = \left\langle \tilde{\psi}_{j,k}, f_{j+1} \right\rangle, \tag{5}$$

where $\tilde{\phi}_{j,k}$ and $\tilde{\psi}_{j,k}$ denote the k-th scaling and wavelet basis functions at level j.

Second Generation Wavelets

Let $\mathbf{x} = (x_k)_{k\in\mathbb{Z}}$ be an infinite sampled signal.

The first operation involved in the lifting procedure is called a *split* or *parity assignment*. To this effect, the signal is divided into two sets, an even set, corresponding to $\mathbf{x}_e = (x_{2k})_{k\in\mathbb{Z}}$ and an odd set, corresponding to $\mathbf{x}_o = (x_{2k+1})_{k\in\mathbb{Z}}$.

The second operation of the lifting procedure is the *prediction step*. This implies approximating the odd samples using the even ones by applying a prediction operator, $\mathbf{P}$, i.e.

$$\mathbf{d} = \mathbf{x}_o - \mathbf{P}\mathbf{x}_e, \tag{6}$$

obtaining the signal $\mathbf{d}$ of *approximation errors* or *detail coefficients*.

Usually, the even and odd sample subsets are highly correlated, and, as a direct consequence, it becomes more efficient to store the signal using the information in $\mathbf{d}$ as it has lower entropy than $\mathbf{x}_o$. The prediction phase is equivalent to mapping $(\mathbf{x}_e, \mathbf{x}_o) \rightarrow (\mathbf{x}_e, \mathbf{d})$. Since $\mathbf{x}_e$ is essentially obtained through a naïve downsampling of the original signal, aliasing occurs and must be appropriately dealt with. This is performed through a third operation, the *update* or *smoothing* phase. Algebraically, this operation is implemented as

$$\mathbf{s} = \mathbf{x}_e + \mathbf{U}\mathbf{d}, \tag{7}$$

where $\mathbf{s}$ is the smoothed downsampled signal and $\mathbf{U}$ is the update operator.

Assembling the *three stages* of the lifting scheme into a sequence, we obtain the flow diagram shown in Fig. 1. The flow allows cascading the resulting filter bank by feeding the even output from one lifting pass to the input of another one, effectively creating a series of coarser approximations of the original signal.

Another immediate property of the lifting design is the straightforward invertibility, i.e.

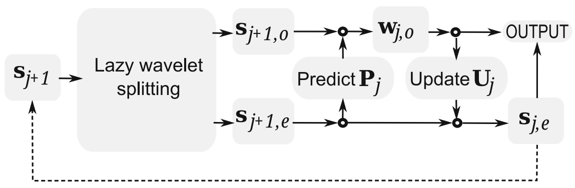

Fig. 1 The *lifting scheme* diagram: Splitting (Eq. (10)) is followed by prediction via Eq. (11) and then by the updating of the low resolution output via Eq. (12)

$$\mathbf{x}_e = \mathbf{s} - \mathbf{U}\mathbf{d} \tag{8}$$
$$\mathbf{x}_o = \mathbf{d} + \mathbf{P}\mathbf{x}_e. \tag{9}$$

Returning to the function space context, we review the mechanism behind the lifting scheme. The first operation, the *split*, when applied to the set of scaling coefficients at level $j+1$ produces a vector of scaling coefficients reindexed such that

$$\mathbf{s}_{j+1} = \begin{bmatrix} \mathbf{s}_{j+1,o} \\ \mathbf{s}_{j+1,e} \end{bmatrix}, \tag{10}$$

where the o and e subscripts stand for *odd* and *even* coefficients, respectively.

We write the *prediction* equation as

$$\mathbf{w}_j = \mathbf{s}_{j+1,o} - \mathbf{P}_j \mathbf{s}_{j+1,e}, \tag{11}$$

and the *update* as

$$\mathbf{s}_j = \mathbf{s}_{j+1,e} + \mathbf{U}_j \mathbf{w}_j, \tag{12}$$

with $\mathbf{P}_j \in \mathbb{R}^{n_{j+1,o} \times n_{j+1,e}}$ being a sparse prediction matrix, $\mathbf{U}_j \in \mathbb{R}^{n_{j+1,e} \times n_{j+1,o}}$ being a sparse update matrix and $n_{j+1,o}$ and $n_{j+1,e}$ being the number of odd and even coefficients, respectively, at level $j+1$. The prediction stage simply exploits the signal redundancy by assuming that the odd coefficients can be estimated as linear combinations of their spatial neighbors. If only the even coefficients are used to approximate the functions in V_{j+1}, then the lost details are compensated for by redistributing them among these remaining coefficients via the update matrix.

Since Eqs. (3)–(5) hold for any f_{j+1}, we can write the predict and update equations for the duals basis functions:

$$\tilde{\Psi}_j^{\mathsf{T}} = \tilde{\Phi}_{j+1,o}^{\mathsf{T}} - \mathbf{P}_j \tilde{\Phi}_{j+1,e}^{\mathsf{T}}, \tag{13}$$
$$\tilde{\Phi}_j^{\mathsf{T}} = \tilde{\Phi}_{j+1,e}^{\mathsf{T}} + \mathbf{U}_j \tilde{\Psi}_j^{\mathsf{T}} = (\mathbf{I} - \mathbf{U}_j \mathbf{P}_j) \tilde{\Phi}_{j+1,e}^{\mathsf{T}} + \mathbf{U}_j \tilde{\Phi}_{j+1,o}^{\mathsf{T}}. \tag{14}$$

To derive similar relations between the primal basis functions, we can start by rewriting Eq. (2) as

$$\begin{bmatrix}\Phi_{j+1,o} & \Phi_{j+1,e}\end{bmatrix}\begin{bmatrix}\mathbf{s}_{j+1,o}\\ \mathbf{s}_{j+1,e}\end{bmatrix} = \begin{bmatrix}\Psi_j & \Phi_j\end{bmatrix}\begin{bmatrix}\mathbf{w}_j\\ \mathbf{s}_j\end{bmatrix}, \tag{15}$$

and, expanding the scaling and wavelet coefficients on the right-hand side using Eqs. (11) and (12), we arrive to

$$\begin{bmatrix}\Phi_{j+1,o} & \Phi_{j+1,e}\end{bmatrix}\begin{bmatrix}\mathbf{s}_{j+1,o}\\ \mathbf{s}_{j+1,e}\end{bmatrix} = \begin{bmatrix}\Psi_j & \Phi_j\end{bmatrix}\begin{bmatrix}\mathbf{s}_{j+1,o} - \mathbf{P}_j\mathbf{s}_{j+1,e}\\ (\mathbf{I} - \mathbf{U}_j\mathbf{P}_j)\mathbf{s}_{j+1,e} + \mathbf{U}_j\mathbf{s}_{j+1,o}\end{bmatrix}. \tag{16}$$

Equation (16) must hold for any combination of the level $j + 1$ lifting coefficients. Hence, let $\mathbf{s}_{j+1,e} = \boldsymbol{\delta}_k$, i.e. the Kronecker unit vector at index k, with $k \in 1 : n_e$, i.e. $\delta_{k,i} = 0, \forall k \neq i$ and $\delta_{k,k} = 1$. Also, let $\mathbf{s}_{j+1,o} = \mathbf{0}$. Evaluating both sides of Eq. (16), we arrive to

$$\Phi_{j+1,e}\delta_k = -\Psi_j\mathbf{P}_j\boldsymbol{\delta_k} + \Phi_j\boldsymbol{\delta_k} - \Phi_j\mathbf{U}_j\mathbf{P}_j\boldsymbol{\delta_k}, \tag{17}$$

or, since this equation holds for any $k \in 1 : n_e$, a more direct formulation can be written as

$$\Phi_{j+1,e} = -\Psi_j\mathbf{P}_j + \Phi_j - \Phi_j\mathbf{U}_j\mathbf{P}_j. \tag{18}$$

By setting $\mathbf{s}_{j+1,e} = \mathbf{0}$ and $\mathbf{s}_{j+1,o} = \delta_k$, with $k \in 1 : n_o$, we find that

$$\Phi_{j+1,o}\delta_k = \Psi_j\delta_k + \Phi_j\mathbf{U}_j\delta_k, \tag{19}$$

or

$$\Phi_{j+1,o} = \Psi_j + \Phi_j\mathbf{U}_j. \tag{20}$$

Right-multiplying both sides of Eq. (20) by $\mathbf{P}_j$ and adding the result to Eq. (18) we obtain:

$$\Phi_j = \Phi_{j+1,e} + \Phi_{j+1,o}\mathbf{P}_j, \tag{21}$$

and then

$$\Psi_j = \Phi_{j+1,o}(\mathbf{I} - \mathbf{P}_j\mathbf{U}_j) - \Phi_{j+1,e}\mathbf{U}_j. \tag{22}$$

Let $\varsigma_j = \int_{-\infty}^{\infty} \Phi_j^{\mathsf{T}}(t)dt$. Then integrating equation (21) yields:

$$\varsigma_j = \varsigma_{j+1,e} + \mathbf{P}_j^{\mathsf{T}}\varsigma_{j+1,o}. \tag{23}$$

Since the integral of the wavelet functions is zero, integrating equation (22) leads to:

$$\mathbf{0} = \varsigma_{j+1,o} - \mathbf{U}_j^{\mathsf{T}}\left(\mathbf{P}_j^{\mathsf{T}}\varsigma_{j+1,o} + \varsigma_{j+1,e}\right) = \varsigma_{j+1,o} - \mathbf{U}_j^{\mathsf{T}}\varsigma_j. \tag{24}$$

The column vectors of $\mathbf{U}_j$ can be retrieved in a one-by-one fashion from Eq. (24). If $\mathbf{u}_{j_k}$ is the kth column vector, then

$$\varsigma_{j+1,o_k} = \mathbf{u}_{j_k}^{\mathsf{T}} \varsigma_{j_k}. \tag{25}$$

It is suggested in [21] to choose the minimum norm solution of this equation as it leads to increased numerical stability, as experimentally shown in [9]. It results that

$$\mathbf{u}_{j_k} = \varsigma_{j+1,o_k} \frac{\varsigma_{j_k}}{\|\varsigma_{j_k}\|^2}. \tag{26}$$

Wavelet Construction for Multivariate Graph Signals

We examine now the changes necessary to go from the above one-dimensional construction to multivariate signals defined on graphs. In general, we are interested in analyzing a function $F(\mathbf{v})$ defined at every vertex $\mathbf{v}$ of the graph. We regard the vertices as having vector coordinates in $\mathbb{R}^n$. The purpose is to find appropriate scaling and wavelet function bases that are independent of the multivariate signal itself, but depend on the graph topology and its edge lengths. Thus, the framework construction we propose must also accommodate both univariate and multivariate functions by treating the latter as tuples of scalar-valued functions. Instead of discussing about time series indices, we now view the graph nodes as means to identify samples where the information is concentrated. The odd-even split is extended to graph vertices, denoting V_{o_l} and V_{e_l} the odd and even subsets at level l, respectively. The notion of parity is irrelevant for graphs, the odd and even denominations serving a labeling purpose. If $\mathbf{v}_{i_l} \in V_{e_l}$ denotes a vertex from the approximation set at level l, we refer to its corresponding scaling vector of coefficients by using the $\mathbf{s}_{l,\mathbf{v}_{i_l}}$ notation. At the highest level of resolution, L, we assimilate the scaling vector to the vertex coordinates, i.e. $\mathbf{s}_{L,\mathbf{v}_{i_L}} := \mathbf{v}_{i_L}^{\mathsf{T}}$. We adopt the same convention for denoting the scaling functions, i.e. $\phi_{l,\mathbf{v}_{i_l}}$, which are scalar valued. The situation is identical for the detail components, which correspond to the odd samples, $\mathbf{v}_{j_l} \in V_{o_l}$, at level l. The detail vector associated with $\mathbf{v}_{j_l}$ will be denoted by $\mathbf{w}_{l,\mathbf{v}_{j_l}}$, while for the wavelet functions the $\psi_{l,\mathbf{v}_{j_l}}$ notation will be adopted. Using these conventions, we can express the equivalent *multiresolution decomposition equation for graphs* as

$$F(\mathbf{v}) = \sum_{l \geq 0} \sum_{\mathbf{v}_o \in V_{o_l}} \mathbf{w}_{l,\mathbf{v}_o} \psi_{l,\mathbf{v}_o}(\mathbf{v}) + \sum_{\mathbf{v}_e \in V_{e_l}} \mathbf{s}_{0,\mathbf{v}_e} \phi_{0,\mathbf{v}_e}(\mathbf{v}). \tag{27}$$

By arranging the scaling and wavelet functions into row vectors, we reproduce Eqs. (21) and (22), establishing the relationship at consecutive resolution levels through the means of prediction and update filters, i.e.

$$\Phi_{l,V_{e_l}} = \Phi_{l+1,V_{e_{l+1}}} + \Phi_{l+1,V_{o_{l+1}}} \mathbf{P}_l, \tag{28}$$

$$\Psi_{l,V_{o_{l+1}}} = \Phi_{l+1,V_{o_{l+1}}} - \Phi_{l,V_{e_l}} \mathbf{U}_l, \tag{29}$$

where $\mathbf{P}_l$ is the $|V_{o_{l+1}}| \times |V_{e_{l+1}}|$ prediction filter matrix and $\mathbf{U}_l$ is the $|V_{e_{l+1}}| \times |V_{o_{l+1}}|$ update filter matrix. Using the above filter matrices, a hierarchical decomposition of the scaling coefficients associated with the vertices of a mesh or a graph can be inferred. These vector-valued coefficients contain both the geometric and attribute coordinates of their corresponding vertices. Let us denote by $\mathbf{s}_{l,V_{e_l}}$ the $|V_{e_l}| \times n$ matrix of scaling coefficients at level l associated with the even nodes, V_{e_l}, the difference vectors computation and even node updates can be written as

$$\mathbf{w}_{l,V_{o_{l+1}}} = \mathbf{s}_{l+1,V_{o_{l+1}}} - \mathbf{P}_l \mathbf{s}_{l+1,V_{e_{l+1}}}, \tag{30}$$

$$\mathbf{s}_{l,V_{e_l}} = \mathbf{s}_{l+1,V_{e_{l+1}}} + \mathbf{U}_l \mathbf{w}_{l,V_{o_{l+1}}}. \tag{31}$$

Equations (30) and (31) describe the analysis stage of critically sampled lifting scheme. It is straightforward to invert this process. This converse operation, the synthesis stage, is translated into the following two equations:

$$\mathbf{s}_{l+1,V_{e_{l+1}}} = \mathbf{s}_{l,V_{e_l}} - \mathbf{U}_l \mathbf{w}_{l,V_{o_{l+1}}}, \tag{32}$$

$$\mathbf{s}_{l+1,V_{o_{l+1}}} = \mathbf{w}_{l,V_{o_{l+1}}} + \mathbf{P}_l \mathbf{s}_{l+1,V_{e_{l+1}}}. \tag{33}$$

Cascading the analysis stages yields a hierarchical wavelet decomposition of the initial mesh. The method stores the intermediary difference vectors $\mathbf{w}_{l,V_{o_{l+1}}}$ and the coarsest level scaling coefficients, $\mathbf{s}_{0,V_{e_0}}$, with $l \in 0 : L$. In order to recover the initial information, the intermediary filter matrices, $\mathbf{P}_l$ and $\mathbf{U}_l$, also need to be stored.

2.2 *Graph-Based Lifting Scheme Operations*

We now proceed to describing the necessary steps for adapting the lifting scheme principles to the irregular graph domain. In doing this, we consider a mechanism for guiding the lazy wavelet partitioning such as salient features loss is reduced during downsampling. We also aim to develop prediction and update filters that minimize the approximation error at lower levels of resolution.

Lazy Wavelet Partitioning

Our choice of a heuristic feature preservation mechanism is the *generalized quadric error metric*, detailed by Garland and Heckbert [17]. The goal of this metric is to facilitate computing the squared distances from any point to the support plane of a triangle. For the remainder of this discussion, we will refer to vertices and their attributes as elements from the $\mathbb{R}^n$ vector space, where the first three components are the geometric coordinates and the remaining $n - 3$ represent the attribute data.

Let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^n$ be column vectors corresponding to the vertices of a triangle and $\mathbf{p} \in \mathbb{R}^n$ an arbitrary point. The core idea of this approach is to algebraically express as a matrix, denoted by $\mathbf{Q}(\Delta(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3))$, the computation of the squared distance from $\mathbf{p}$ to the support plane of these 3 vertices. It is then easy to compute the sum of squared distances from **p** to the support planes of a triangle family, $(\Delta_i(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3))_{i=1:N}$, as

$$\sum_{i=1:N} d(\mathbf{p}, \Delta_i(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3))^2 = \begin{pmatrix}\mathbf{p}\\1\end{pmatrix}^{\mathsf{T}} \left\{ \sum_{i=1:N} \mathbf{Q}\left(\Delta_i(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)\right) \right\} \begin{pmatrix}\mathbf{p}\\1\end{pmatrix}. \tag{34}$$

In the original incremental simplification algorithm [17], the set of faces in the one-ring neighborhood of each vertex is used to compute an associated matrix

$$\mathbf{Q}(\mathbf{v}) = \sum_{\Delta_k \in \mathcal{N}_t^1(\mathbf{v})} \mathbf{Q}(\Delta_k), \tag{35}$$

where $\mathcal{N}_t^1(\mathbf{v})$ represents the set of all triangles incident at **v**.

The advantage of using the matrix notation is manifested when performing edge collapses and fusing the endpoint vertices. Whenever two vertices, $\mathbf{v}_a$ and $\mathbf{v}_b$, are replaced by a new vertex, **w**, the local geometric information that was characterized by the neighborhoods of these vertices is preserved by setting $\mathbf{Q}(\mathbf{w}) \leftarrow \mathbf{Q}(\mathbf{v}_a) + \mathbf{Q}(\mathbf{v}_b)$. This way, although the mesh is coarser, the new vertex still retains the local geometric variability of the initial model. Thus, the history of collapses is added together and represented as a single quadric matrix.

The matrix terms in Eq. (35) describe quadrics in the sense that all isosurfaces obtained from varying point **p** in Eq. (34) are quadrics. The term *quadric error metric* is thus justified since these matrices offer a means of estimating an error measure from an arbitrary position **p** to a local patch around any vertex **v**. As described in [7], this metric also allows for the introduction of a cost function associated to each vertex:

$$\texttt{cost}(\mathbf{v}) = \begin{pmatrix}\mathbf{v}\\1\end{pmatrix}^{\mathsf{T}} \left(\sum_{\mathbf{v}_i \in \mathcal{N}_v^1(\mathbf{v})} \mathbf{Q}(\mathbf{v}_i) \right) \begin{pmatrix}\mathbf{v}\\1\end{pmatrix}, \tag{36}$$

where $\mathcal{N}_v^1(\mathbf{v})$ denotes the direct neighbors of **v**, or the *one-ring* vertex set, as represented in Fig. 2.

In [7] we introduced an additional saliency measurement attribute and treat the vertices of the model as points in $\mathbb{R}^{n+1}$. We opt for a discrete bending energy (or thin plate energy) estimation since it encompasses curvature and area information and it is also an isometric invariant.

In the continuous case, the bending energy is a well-defined quantity. If the principal curvatures can be computed over a surface patch A, then the amount of elastic potential energy stored in that region can be computed by evaluating the integral

$$E_b = \int_A \left(\kappa_1^2 + \kappa_2^2\right) dA, \tag{37}$$

where κ_1 and κ_2 are the principal curvature functions defined over the patch A.

Algorithm 1 Thin plate energy computation

```
INPUT: the mesh $M = (V, E)$
OUTPUT: the bending energy of the vertex set, $E_{bending}(V)$
for $\mathbf{v}_i \in V$ do
  COMPUTE: the Gaussian and mean curvatures of $\mathbf{v}_i$, $K(\mathbf{v}_i)$ and $H(\mathbf{v}_i)$
  COMPUTE: $f(\mathbf{v}_i) = 4H(\mathbf{v}_i)^2 - 2K(\mathbf{v}_i)$
end for
for $\mathbf{v}_i \in V$ do
  $E_{bending}(\mathbf{v}_i) = 0$
  for $\Delta \in \mathcal{N}_f^1(\mathbf{v}_i)$ do
    COMPUTE: $E_{bending}(\mathbf{v}_i) += \int_\Delta f dA$
  end for
end for
```

The evaluation of the discrete Gaussian curvature, denoted as $K(\mathbf{v})$, and of the mean curvature, denoted as $H(\mathbf{v})$, can be performed as suggested by [28]. Although such estimates are known to be sensitive to noise, the data typically resulting from LiDAR sets do not exhibit irregularities that could affect the robustness. Regardless of this drawback, a curvature-based energy represents a natural measure for local geometric saliency over a one-ring neighborhood. In [9, 10] we further discuss potential alternatives for use on data heavily affected by noise.

To evaluate the discrete version of integral (37), we can use the Gaussian curvature, K, and the mean curvature, H, to compute the sum of squared principal curvatures as $\kappa_1^2 + \kappa_2^2 = 4H^2 - 2K$. The discrete counterpart of this integral then provides an estimate for the discrete bending energy concentrated at each vertex $\mathbf{v}_i$ and is computed over its one-ring neighborhood. We evaluate the discretized bending energy near a vertex $\mathbf{v}_i$ as

$$E_b(\mathbf{v}_i) = \sum_{\Delta(\mathbf{v}_i,\mathbf{v}_j,\mathbf{v}_k)\in\mathcal{N}_f^1(\mathbf{v}_i)} \frac{f(\mathbf{v}_i) + f(\mathbf{v}_j) + f(\mathbf{v}_k)}{6} \cdot \left\| (\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i) \right\|, \tag{38}$$

where $f \equiv (\kappa_1^2 + \kappa_2^2)$ and $\mathcal{N}_f^1(\mathbf{v}_i)$ is the set of all incident triangles, denoted by $\Delta(\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k)$, at $\mathbf{v}_i$.

We summarize the calculation process of the discrete version of Eq. (37) in Algorithm 1.

To include the computed bending energy in the cost function from Eq. (36), one can substitute a vertex $\mathbf{v}$ with an $(n+1)$-dimensional one, $\bar{\mathbf{v}} = \begin{pmatrix} \mathbf{v} \\ E_{bending}(\mathbf{v}) \end{pmatrix}$.

The purpose of computing the per-vertex cost values is to establish an importance-based ordering of this set. The greedy strategy we employ to label all vertices as either even or odd is summarized in Algorithm 2. The labeling process is iterative and marks the vertices intuitively, according to their computed importance. During each iteration, the vertex having the lowest cost is extracted from the set of unmarked vertices. Its one-ring neighbors are then colored as even, while the extracted sample

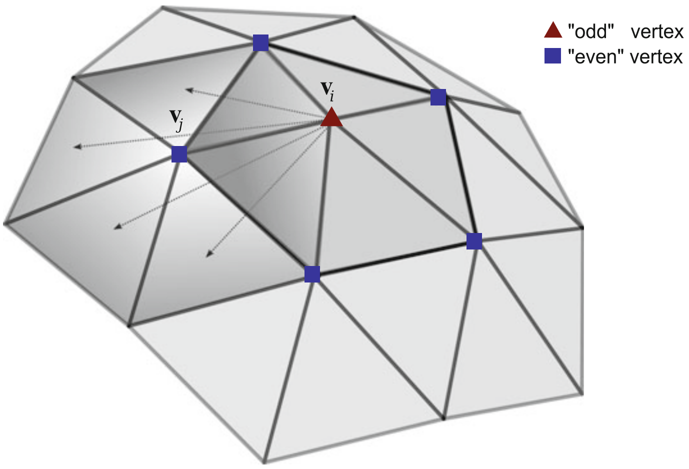

Fig. 2 The cost of removing $\mathbf{v}_i$ with respect to its one-ring neighbors. Each $\mathbf{v}_j \in \mathcal{N}_v^1(\mathbf{v}_i)$ contributes the sum of distances from $\mathbf{v}_i$ to the support planes of each incident face at $\mathbf{v}_j$

Algorithm 2 Vertex labeling and remeshing algorithm

```
INPUT: the mesh $M_{in} = (V, E_{in})$
OUTPUT: an *odd-even* partitioning, $V = V_o \cup V_e$, coarse mesh topology $M_{out} = (V_e, E_{out})$
COMPUTE: for each $\mathbf{v} \in V$, compute $E_{bending}(\mathbf{v})$ via Algorithm 1
COMPUTE: for each $\mathbf{v} \in V$, compute $\texttt{cost}(\bar{\mathbf{v}})$ as defined in (36), where $\bar{\mathbf{v}} = \left(\mathbf{v}^\mathsf{T},\ E_{bending}(\mathbf{v})\right)^\mathsf{T}$
SORT: $V^* = \texttt{sort}(V + in)$ using $\texttt{cost}(\bar{\mathbf{v}})$ as a key
ASSIGN: $V_e = \emptyset$, $V_o = \emptyset$, $E_{out} = E_{in}$
while $V^* \neq \emptyset$ do
  $\mathbf{v} = \arg\min_{\mathbf{v} \in V^*} (\texttt{cost}(\bar{\mathbf{v}}))$
  if $\texttt{can\_triangulate}(\mathcal{N}_v^1(\mathbf{v}) \setminus \{\mathbf{v}\})$ then
    for $\mathbf{v}_i \in \mathcal{N}_v^1(\mathbf{v})$ do
      $E_{out} = E_{out} \setminus \{\overline{\mathbf{v}, \mathbf{v}_i}\}$
    end for
    $E_{out} = E_{out} \cup \texttt{create\_edges}(\mathcal{N}_v^1(\mathbf{v}))$
    $V_o = V_o \cup \{\mathbf{v}\}$
    $V_e = V_e \cup \mathcal{N}_v^1(\mathbf{v})$
    $V^* = V^* \setminus \left(\{\mathbf{v}\} \cup \mathcal{N}_v^1(\mathbf{v})\right)$
  else
    $V_e = V_e \cup \{\mathbf{v}\}$
    $V^* = V^* \setminus \{\mathbf{v}\}$
  end if
end while
```

is marked as odd, as depicted in Fig. 2. The intuition behind the process reflects the goal of removing less relevant samples, from high redundancy areas.

Analyzing the vertex classification and remeshing performed by Algorithm 2, we notice the constraint of marking the neighbors of a removed odd node as even. This is justified by the need to consistently separate the two vertex classes, a practice

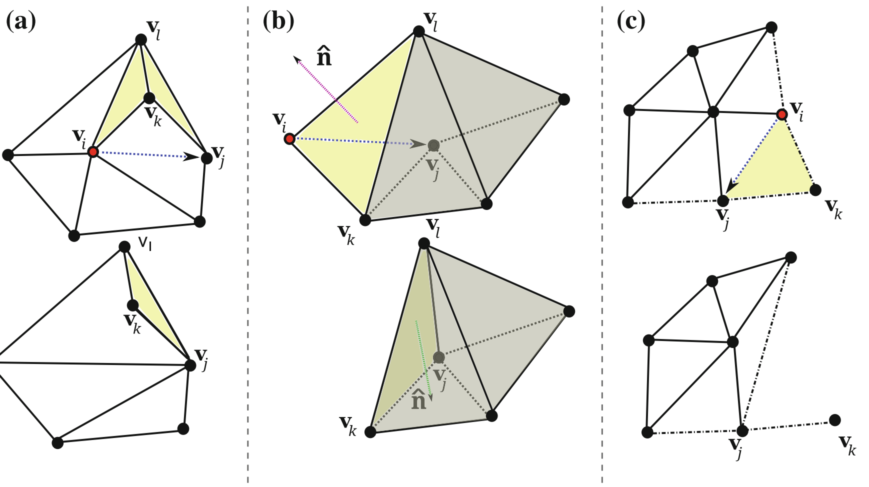

Fig. 3 Illegal collapse situations where the highlighted faces are involved in a breaching of one or several criteria of the `can_triangulate` function from Algorithm 2. In case **a**, the collapse "welds" together two faces, back-to-back, introducing an edge common to more than two triangles. In situation **b**, the highlighted face is "flipped" since the normal unit vectors before and after the collapse point in opposite directions. "Cutting a corner" is the operation depicted in situation **c**, where the corner vertex is no longer adjacent to at least two edges

also employed by [18]. Another key property of this algorithm is the partial graph connectivity alteration incurred by the removal of the odd samples. Together with these nodes, their adjacent edges are also removed. Whether or not the one-ring hole bounded by $\mathscr{N}_v^1(\mathbf{v}) \setminus \{\mathbf{v}\}$ can be triangulated is also a key factor in deciding the label of $\mathbf{v}$. The `can_triangulate(·)` function is implemented as a set of geometrical and topological consistency criteria that must be satisfied by a valid triangulation. This problem has numerous solutions, one example being the *splitting plane* method of [32] known to produce more regular aspect ratio triangulations. It is possible for a vertex having the lowest importance score to produce invalid triangulations, and, in this case, the vertex is marked as even. In our implementation, the triangulation of the one-ring hole is easily performed using half-edge collapses. A valid edge is selected from the set of all edges adjacent to the removed odd vertex such that the quadric error measured using the $\mathbf{Q}(\mathbf{v}_e)$ matrix of its even endpoint is the smallest of all other collapsible pairs. We refer to a vertex pair to be collapsible if it is connected through an edge and if by translating the odd vertex over the even one the discrete manifold property of the mesh is not affected and no newly created triangular face is flipped with respect to its original normal vector (refer to Fig. 3 for several examples of illegal collapses). Additionally, to prevent boundary shrinkage, the odd boundary vertices cannot be translated over interior even ones. Boundary vertices can be part of a collapse if the edge connecting them is a boundary one as well. We also allow for merging an interior vertex with a boundary one when the collapse direction is from the interior towards the border.

Prediction Filter Construction

The goal of the prediction operation that follows the even-odd splitting is to compute estimates for the odd nodes of the graph from their even neighboring nodes. Intuitively, if the predicted values are closer to the actual odd samples, it is possible to recover a faithful representation of the entire graph by storing only a subset of the initial data. The resulting estimates are usually obtained by applying a low-pass filter to the even subset, while computing the difference between the estimates and the actual odd samples resembles the behavior of a high-pass filter. As a graph low-pass filter, we propose using a discrete Laplacian operator, given its smoothing and averaging effects. In many applications, Laplacian filters are also used for suppressing local irregularities (both features and noise). Thus, a similar effect could be achieved by removing the ensuing details.

To understand how the prediction filter weights are computed, we recall the concept of a graph Laplacian operator. For a weighted graph, the Laplacian matrix is obtained as the difference between the weighted diagonal degree matrix, $\mathbf{D} = (d_{i,i}) = \sum_j \omega_{i,j}$, and the cost matrix, $\boldsymbol{\Omega} = (\omega_{i,j})$, where $\omega_{i,j}$ are the weights associated with the $(\mathbf{v}_i, \mathbf{v}_j)$ edges. Thus, if $\mathbf{L} = \mathbf{D} - \boldsymbol{\Omega}$, the random-walk Laplacian operator is defined as

$$\mathscr{L}_{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\boldsymbol{\Omega}. \tag{39}$$

The geometric interpretation for the action this operation has on a mesh is a smoothing effect, also achievable through the use of a generalized umbrella operator. An in-depth comparative discussion of various Laplacian discretizations is offered by [38]. Concretely, the extraction of the difference vectors $\mathbf{w}_{l,V_{o_{l+1}}}$ from Eq. (30) is similar to a Laplacian smoothing where the difference between the smoothed vertex and its actual position is stored for later reference. For a single vertex, $\mathbf{v}_n \in V_{o_{l+1}}$, this equation can be rewritten as

$$\mathbf{w}_{l,\mathbf{v}_n} = \mathbf{s}_{l+1,\mathbf{v}_n} - \sum_{\mathbf{v}_m \in \mathscr{N}_v^1(\mathbf{v}_n)} p_{l,\mathbf{v}_n}(\mathbf{v}_m)\mathbf{s}_{l+1,\mathbf{v}_m}, \tag{40}$$

where $p_{l,\mathbf{v}_n}(\mathbf{v}_m)$ is the prediction weight coefficient at level l associated with vertex $\mathbf{v}_n$ and contributed by one of its even, one-ring neighbors, $\mathbf{v}_m$. Depending on the Laplacian discretization, several prediction weight choices are possible. In general, the prediction weights in Eq. (40) are computed as

$$p_{l,\mathbf{v}_n}(\mathbf{v}_m) = \frac{\omega_{n,m}}{\sum_{\mathbf{v}_k \in \mathscr{N}_v^1(\mathbf{v}_n)} \omega_{n,k}}. \tag{41}$$

One of the more popular Laplacian design choices is the *cotangent weights* Laplacian, computed as described by Meyer et al. [28]. Abdul Rahman et al. [1] recommend this Laplacian as a prediction filter for the analysis of free-form surfaces with additional attributes. Although this filter is suitable for smoothing tasks, its weights depend strictly on the geometric information and not also on any additional attributes. A simple alternative for this operator, that introduces movements in the tangential plane of a vertex, is the Fujiwara or geometric Laplacian [16], with weights defined as

$$\omega_{i,j} = {l_{i,j}}^{-1}, \tag{42}$$

where $l_{i,j}$ is the length of the $(\mathbf{v}_i, \mathbf{v}_j)$ edge. This operator is scale dependent and it preserves the distribution of triangle sizes. The main advantages of this design are: its dependency on all geometry and range attributes, its smoothing effect being closer to the cotangent weight formulation than that of the umbrella operator, and the predicted point being shifted towards its closest neighbors, thus inherently providing a better approximation.

An alternative to the Laplacian prediction filter is to employ a least squares fitting of the weights in order to minimize the approximation error. This approach, adopted in several works [27, 37], proved to be numerically unstable when directly applied to terrain sets. To overcome this issue, Wagner et al. [37] did not include the boundary vertices in the downsampling process. In regions where the one-ring neighborhood does not have a convex (x, y) projection, negative weights can appear. Furthermore, the large magnitude of the weights may lead to numerical instability during the subsequent update stage. To counteract these effects, we propose a non-negative least squares (NNLS) fitting of the weights in Eq. (40), such that the magnitude of the $\mathbf{w}_{l,\mathbf{v}_n}$ vector is minimized. To achieve a similar Laplacian smoothing effect, the sum $\sum_{\mathbf{v}_m \in \mathcal{N}_v^1(\mathbf{v}_n)} p_{l,\mathbf{v}_n}(\mathbf{v}_m)$ should be equal to 1. This constraint can be directly added to the NNLS solver. By design, this modification improves the root mean square error throughout the hierarchical downsampling, as it will be later discussed in the results section. We note that positive and convex weights allow for the removal of odd boundary vertices. Nevertheless, computing these weights incurs a computational penalty, so we only recommend this choice for scenarios where minimizing the approximation error is crucial.

In terms of storage complexity, the Laplacian matrix is very sparse. As a consequence, the prediction matrix at level l, $\mathbf{P}_l$, is also sparse, each of its rows being populated with the weights used to predict the same odd vertex, $\mathbf{v}_n$, from its even neighbors.

Update Filter Construction

The heuristically guided lazy wavelet removal of details minimizes feature loss, but does not propagate detail loss to inferior levels. Wavelet transforms manage this problem by redistributing the extracted detail among the coarser scales. This information is effectively contained in the difference vectors. Without compensating for these losses, the algorithm would mostly resemble incremental simplification. The update filter of the lifting scheme is responsible for distributing the lost details among the even vertices in a way that preserves an aggregate signal property such as signal average. We opt for this choice because it helps maintain both overall shape and decreases the approximation error, as we will later experimentally observe.

Because the prediction filter determines the amount of detail loss, the update filter should express a dependency on the prediction weights. In [27], the authors suggest the following expression for the update filter vectors:

$$\mathbf{u}_{l,\mathbf{v}_u} = \frac{0.5}{\sum\limits_{\mathbf{v}_{p,i} \in \mathcal{N}_v^1(\mathbf{v}_u) \cap V_{o_{l+1}}} p_{l,\mathbf{v}_{p,i}}} \left[p_{l,\mathbf{v}_{p,1}}, p_{l,\mathbf{v}_{p,2}}, \ldots, p_{l,\mathbf{v}_{p,k}} \right], \tag{43}$$

where $\mathbf{v}_u$ represents an even vertex, $\mathbf{v}_{p,i}$ denotes an odd one-ring neighbor of this vertex, and $p_{l,\mathbf{v}_{p,i}} \equiv p_{l,\mathbf{v}_{p,i}}(\mathbf{v}_u)$ is the prediction weight $\mathbf{v}_u$ contributed in estimating its $\mathbf{v}_{p,i}$ odd neighbor. By using this design, there is no guarantee the signal average will be preserved, unless applied to unweighted graphs, hence when using the umbrella operator Laplacian.

Abdul-Rahman et al. [1] proposed a similar approach where the update filter construction aims to directly preserve the average value of the one-ring neighborhood of an odd vertex before and after its removal. Using the same update vector for all even neighbors, the following equation ensues:

$$\frac{1}{N+1}\left(\mathbf{w}_{l,\mathbf{v}_n} + \sum_{\mathbf{v}_m \in \mathcal{N}_v^1(\mathbf{v}_n)} (p_{l,\mathbf{v}_n}(\mathbf{v}_m) + 1)\mathbf{s}_{l+1,\mathbf{v}_m} \right) = \tag{44}$$

$$\frac{1}{N} \sum_{\mathbf{v}_m \in \mathcal{N}_v^1(\mathbf{v}_n)} (\mathbf{s}_{l+1,\mathbf{v}_m} + \mathbf{u}_{l,\mathbf{v}_m}), \tag{45}$$

with $N = |\mathcal{N}_v^1(\mathbf{v}_n)|$, and the update vectors $\mathbf{u}_{l,\mathbf{v}_i} = \mathbf{u}_{l,\mathbf{v}_j} = \mathbf{u}_{\mathcal{N}_v^1(\mathbf{v}_n)}$ for any $\mathbf{v}_i, \mathbf{v}_j \in \mathcal{N}_v^1(\mathbf{v}_n)$. Thus, the uniform update vector is determined as

$$\mathbf{u}_{\mathcal{N}_v^1(\mathbf{v}_n)} = \frac{1}{N+1}\mathbf{w}_{l,\mathbf{v}_n} + \sum_{\mathbf{v}_m \in \mathcal{N}_v^1(\mathbf{v}_n)} \frac{N p_{l,\mathbf{v}_n}(\mathbf{v}_m) - 1}{N(N+1)} \mathbf{s}_{l+1,\mathbf{v}_m}. \tag{46}$$

In case the prediction filter weights correspond to the umbrella operator type of Laplacian, the second term in Eq. (46) vanishes. Generally, this design requires storing update weights for both difference vectors and even nodes from the previous level, thus becoming a more memory consuming approach. By aiming to directly preserve the one-ring average of the scaling coefficients, it becomes more difficult to find update weights that depend only on the difference vectors. This is due to the fact that determining such weights implies solving a sparse system involving all scaling coefficients. In the multivariate scenario, this system becomes overdetermined and an exact solution may not exist. In this situation, an approximate solution must be searched for. Overall, this process is more complex than the entire lifting pipeline.

In [9] we also considered the solution proposed by Jansen et al. [22]. In principle, it also exploits the average preserving requirement. This prerequisite provides a mathematical constraint that can be expressed in terms of the integrals of the scaling functions, as described for the one-dimensional context discussion. Let $\varsigma_{l,\mathbf{v}_u}$ be the integral of the scaling function $\phi_{l,\mathbf{v}_u}$ (i.e. an element of $\Phi_{l,V_{e_{l+1}}}$ corresponding to the vertex $\mathbf{v}_u \in E_l$). Rewriting Eq. (23), we obtain

$$\varsigma_{l,\mathbf{v}_u} = \varsigma_{l+1,\mathbf{v}_u} + \sum_{\mathbf{v}_k \in \mathscr{N}_v^1(\mathbf{v}_u) \cap V_{o_{l+1}}} p_{l,\mathbf{v}_k}(\mathbf{v}_u)\varsigma_{l+1,\mathbf{v}_k}. \tag{47}$$

The wavelet biorthogonality condition requires for each wavelet function to have zero integral. In consequence, when integrating equation (29) the left hand side term vanishes, i.e.,

$$0 = \varsigma_{l+1,V_{o_{l+1}}} - \mathbf{U}_l \varsigma_{l,V_{e_l}}. \tag{48}$$

Further rewriting Eq. (25) for each predicted vertex, $\mathbf{v}_p \in V_{o_{l+1}}$, leads to

$$\varsigma_{l+1,\mathbf{v}_p} = \mathbf{u}_{l,\mathbf{v}_p}^{\mathsf{T}} \varsigma_{l,\mathscr{N}_v^1(\mathbf{v}_p)}, \tag{49}$$

where $\mathbf{u}_{l,\mathbf{v}_p}$ is the vector containing the update weights this vertex contributes with for each of its one-ring, even vertices. Finding $\mathbf{u}_{l,\mathbf{v}_p}$ requires solving an overdetermined equation of the form $\alpha = \mathbf{u}^{\mathsf{T}}\mathbf{v}$, where $\mathbf{u}$ is the unknown vector. From all possible solutions, both Jansen et al. [22] and Wagner et al. [37] choose the minimum update norm solution due to its stabilizing effect. This translates into setting $\mathbf{u} = \frac{\alpha \mathbf{v}}{\|\mathbf{v}\|^2}$. Finally, Eq. (26), which gives the expression of the update vector of coefficients for $\mathbf{v}_p$, becomes

$$\mathbf{u}_{l,\mathbf{v}_p} = \frac{\varsigma_{l+1,\mathbf{v}_p}}{\sum_{\mathbf{v}_k \in \mathscr{N}_v^1(\mathbf{v}_p)} \varsigma_{l,\mathbf{v}_k}^2} \varsigma_{l,\mathscr{N}_v^1(\mathbf{v}_p)}. \tag{50}$$

With these results, the update Eq. (31) of an even scaling coefficient for a vertex $\mathbf{v}_u$ is written as

$$\mathbf{s}_{l,\mathbf{v}_u} = \mathbf{s}_{l+1,\mathbf{v}_u} + \sum_{\mathbf{v}_n \in \mathscr{N}_v^1(\mathbf{v}_u) \cap V_{o_{l+1}}} u_{l,\mathbf{v}_u}(\mathbf{v}_n)\mathbf{w}_{l,\mathbf{v}_n}. \tag{51}$$

The choice of the scaling functions does not affect the so far described transform. Since the initial integrals $\varsigma_{l+1,E_{l+1}}$ need to be computed, one option is to assume a choice of the scaling functions such that these integrals are all equal to 1.

Given the fact that a terrain mesh has sparse connectivity, the filter matrices are also sparse. Each odd node is surrounded by even nodes, ensuring a 25% average reduction factor for each decimation stage. Asymptotically, almost 80% of the initial edge density will still be required to store the filter matrices. Nevertheless, the information is still very sparse and lends itself for specific algebraic manipulations after the cascaded analysis stages.

Algorithm 3 Downsampling algorithm

```
INPUT: a $d$-dimensional point cloud consisting of vertex-attribute pairs.
OUTPUT: the lowest resolution vertex set, $\mathbf{s}_{0,V_0}$, the chain of prediction and update matrices, $(\mathbf{P}_l)_l$ and $(\mathbf{U}_l)_l$, and the string of difference vectors, $(\mathbf{w}_{l,V_{o_{l+1}}})_l$, where $l$ ranges from 0 to the total number of refinement levels, $L$.
TRIANGULATE: the input set, $V_L$.
for l := L - 1 downto 0 do
    COMPUTE: the removal cost of each vertex using algorithm 1
    PARTITION: the vertex set $V_{l+1} = V_{e_{l+1}} \cup V_{o_{l+1}}$ using Algorithm 2.
    COMPUTE: $\mathbf{w}_{l,V_{o_{l+1}}}$ and $\mathbf{s}_{l,V_{e_l}}$ from $\mathbf{s}_{l+1,V_{e_{l+1}}}$ using Eqs. (30) and (31)
    STORE: $\mathbf{P}_l$, $\mathbf{U}_l$ and $\mathbf{w}_{l,V_{o_{l+1}}}$
end for
STORE: $\mathbf{s}_{0,V_0}$
```

2.3 Algorithm Overview

The lifting scheme flow depicted in Fig. 1 is at the core of the iterative downsampling process, illustrated in Fig. 4 and summarized in Algorithm 3. In overview, we can identify three main stages: the cost computation, the labeling or *lazy wavelet* decomposition and the analysis itself. Together, these stages resemble the structure of a classical filter bank.

This algorithm has $O(N \log N)$ complexity (where N is the total number of vertices). This higher complexity is due to the nodes being sorted according to their removal cost. While an $O(N)$ complexity is achievable, the salient features will not

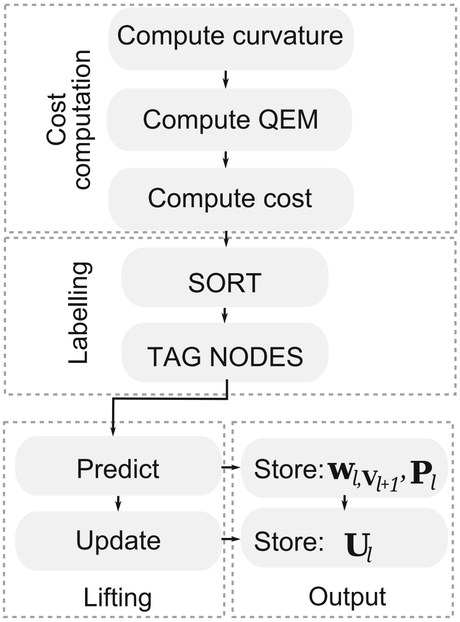

Fig. 4 An overview of our hybrid algorithm for downsampling an input set

be preserved as faithfully (see Fig. 8). The initial triangulation can be performed only once, during the pre-processing stage. During the downsampling steps, the one-rings of the odd vertices can be re-triangulated on the fly, keeping a consistent manifold connectivity.

The spatial complexity of this method is linear with respect to the size of the input data set. More precisely, the lowest resolution data and the difference vectors require the same amount of storage as the initial point cloud. Empirically, the triangulated meshes will not be far from semi-regular, thus the prediction weights will require $6N$ scalars in total. For the update weights, sparse structures corresponding to even vertices being surrounded by 3 odd vertices on average have to be stored. Assuming a decimation rate of 25%, the spatial requirements for storing the update weights asymptotically approach $15N$.

Quadric Error Matrix Update Procedure

The quadric error metric matrix fusion of collapsed vertex pairs is a property that we adapt locally, in the odd vertex neighborhood. We achieve this by distributing the matrix corresponding to an odd sample, $\mathbf{Q}(\mathbf{v}_{o_{j+1}})$, among its even neighbors. Thus, each even matrix is updated by

$$\mathbf{Q}(\mathbf{v}_{e_j}) = \mathbf{Q}(\mathbf{v}_{e_{j+1}}) + \sum_{\mathbf{v}_{o_{j+1}} \in \mathscr{N}_v^1(\mathbf{v}_{e_{j+1}}) \cap V_{o_{j+1}}} \rho_{j,\mathbf{v}_{e_{j+1}}}(\mathbf{v}_{o_{j+1}}) \mathbf{Q}(\mathbf{v}_{o_{j+1}}), \tag{52}$$

where $\rho_{j,\mathbf{v}_{e_{j+1}}}(\mathbf{v}_{o_{j+1}})$ is the redistribution weight describing the influence of $\mathbf{v}_{o_{j+1}}$ on its even neighbor, $\mathbf{v}_{e_{j+1}}$. We propose using the same weights that the prediction filter relies on estimating the lost information, i.e.,

$$\rho_{j,\mathbf{v}_{e_{j+1}}}(\mathbf{v}_{o_{j+1}}) = \frac{p_{j,\mathbf{v}_{o_{j+1}}}(\mathbf{v}_{e_{j+1}})}{\sum_{\mathbf{v}_{u_{j+1}} \in \mathscr{N}_v^1(\mathbf{v}_{o_{j+1}})} p_{j,\mathbf{v}_{o_{j+1}}}(\mathbf{v}_{u_{j+1}})}. \tag{53}$$

This choice is natural since the even vertex that contributes more to the prediction of an odd neighbor receives a larger fraction of its quadric error matrix. We justify this to be a more natural choice given that most prediction weights automatically encode a node similarity magnitude. Other quadric error matrix redistribution weights choices and strategies are possible and were analyzed in [7, 8]. Experimentally, the choice of weights presented through Eq. (53) was determined to achieve, on average, slightly more accurate approximations than the alternatives presented in [8].

3 Results and Discussion

We assessed the validity and efficiency of our method through a series of experiments involving three different LiDAR sets. The first model is a scan fragment of the Great Smoky Mountains (available through the www.opentopography.org/ portal) with a

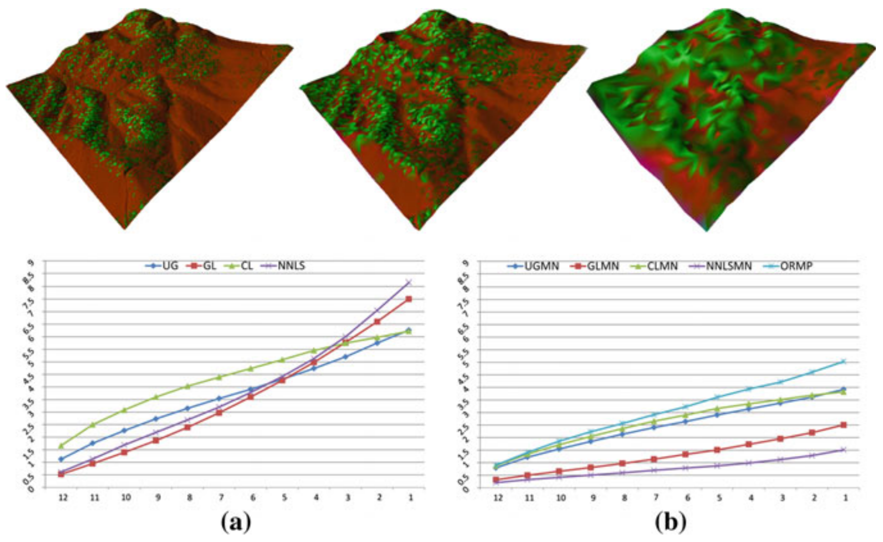

Fig. 5 Smoky Mountains fragment. Top row, left to right: shape and attribute evolution at levels 0, 12 and 24 using the UG filters. Bottom row: comparative charts for the RMSE evolution across 12 analysis sequences

density of approximately 2.23 points per square meter and a size of 280,000 points (Fig. 5). The second set is larger and much denser at approximately 20 points per square meter and a count of 9 million (Fig. 6). The third set is larger but less denser, consisting of 11.5 million points, at a density of 5 points per square meter (Fig. 7). These larger and denser LiDAR samples were acquired through custom aerial scans conducted over two regions of the Romanian Carpathians. All terrain samples contain both geometry (point coordinates) and attribute information (i.e., vegetation type, as a scalar between 0 and 20, and height above ground, as a scalar between 0 and 30). All coordinates are translated and scaled to fit within a zero-centered unit bounding box dividing them by their range width. This way we significantly alleviate the effect of the diverse scales on the final outcome. However, the attributes could be scaled differently if one desires to diminish or enhance their contribution.

3.1 Root Mean Squared Error Measurements

As a measure of quality and accuracy, we have chosen the root mean square error (RMSE). Although other error measuring mechanisms exist, the RMSE or L_2 norm is one of the simplest and most efficient indicators of shape and attribute quality (see [30] for a more in-depth discussion of the properties and applications of this error). In our case, the RMSE computation concerns both geometry and attribute vertex coordinates, except for the artificially added bending energy. Thus, if $\mathbf{s}_{l,\mathbf{v}_i}$ is

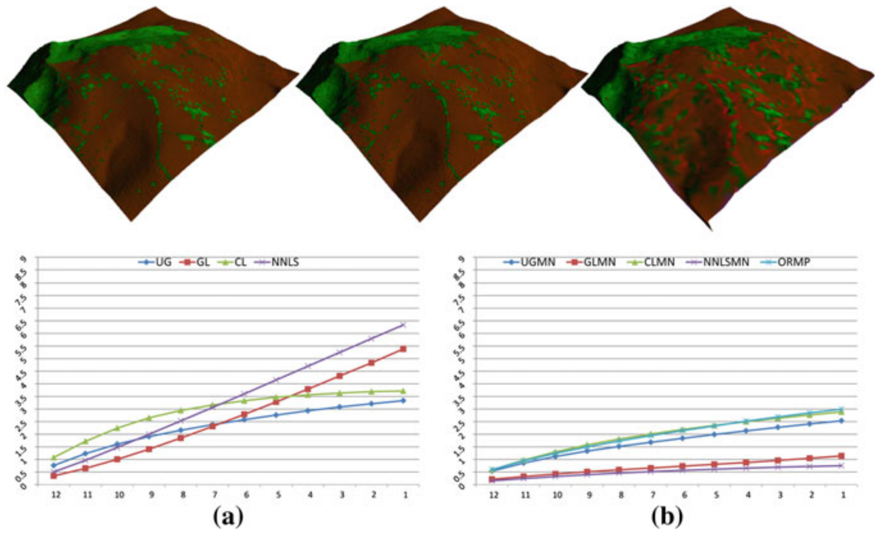

Fig. 6 High-density Carpathian Mountains fragment. Top row, left to right: shape and attribute evolution at levels 0, 12 and 24 the UG filters. Bottom row: comparative charts for the RMSE evolution across 12 analysis sequences

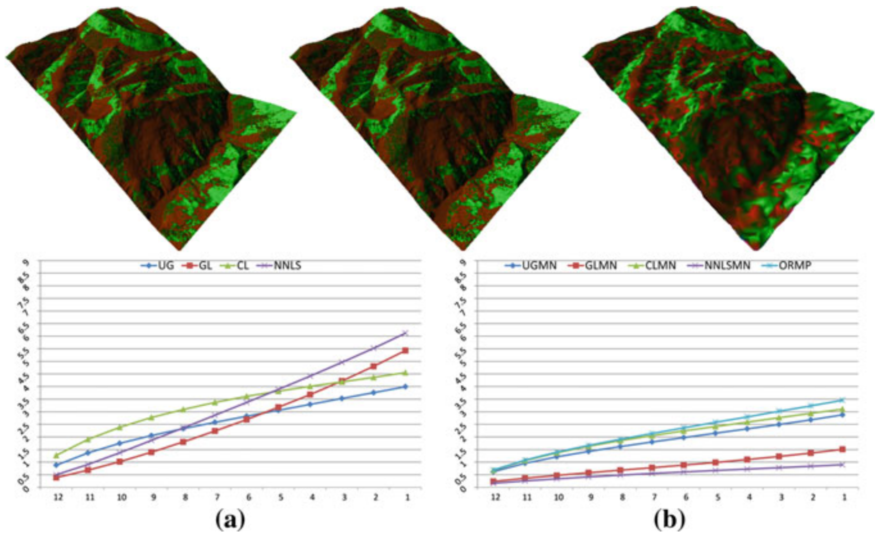

Fig. 7 Low-density Carpathian Mountains fragment. Top row, left to right: shape and attribute evolution at levels 0, 12 and 24 the UG filters. Bottom row: comparative charts for the RMSE evolution across 12 analysis sequences

the scaling coefficient at level l of a node $\mathbf{v}_i$, and $\mathbf{s}_{L,\mathbf{v}_i}$ is the corresponding coefficient in the initial input set, the RMSE is evaluated as

$$\texttt{RMSE}_l = \sqrt{\sum_{\mathbf{v}_i \in E_l} \|\mathbf{s}_{L,\mathbf{v}_i} - \mathbf{s}_{l,\mathbf{v}_i}\|^2}. \tag{54}$$

For the multiresolution representation experiments, we have also considered several prediction and update filter designs. In this respect, we labeled the result sets according to the type of prediction and update filters as follows:

- *prediction filters*: uniform weights obtained by using the unweighted graph Laplacian (UG), present in the design of Martinez and Ortega [27], the cotangent Laplacian weights (CL), chosen by Abdul-Rahman et al. [1], the Fujiwara or geometric Laplacian (GL), which is our proposal for a Laplacian design for multivariate data and a constrained non-negative least squares weights design (abbreviated as NNLS), which we propose in order to minimize the detail vector norms.
- *update filters*: the filter design proposed by Martinez and Ortega [27] compensates for the detail loss incurred by the removal of the odd vertices, but it does not preserve the graph signal average for weighted graphs, unless all weights are uniform. On the other hand, the design used in [22, 37], achieves this goal while minimizing the norm of the update coefficient vectors. To distinguish between the first option and the minimum norm update weights, we have added the MN suffix to the prediction filter abbreviation. The third and final design option, proposed in [1], preserves the local, one-ring mean of the data samples and is abbreviated as ORMP.

3.2 *Feature Selectivity and Robustness to Noise*

For a proof of concept, we subjected the Smoky Mountains terrain fragment to a sequence of 8 analysis steps. The same set was again analyzed using the lazy-wavelet partitioning strategy employed in [27, 37]. Without our proposed feature preservation heuristic, the lowest resolution representation will lose a bigger fraction of the sharp features, as depicted in Fig. 8.

Computing the discrete curvatures using the method of [28] is recommended for meshes with low to no noise. In [9, 10], we have experimentally shown that the proposed heuristic partitioning of the data points into odd and even categories is robust to noise.

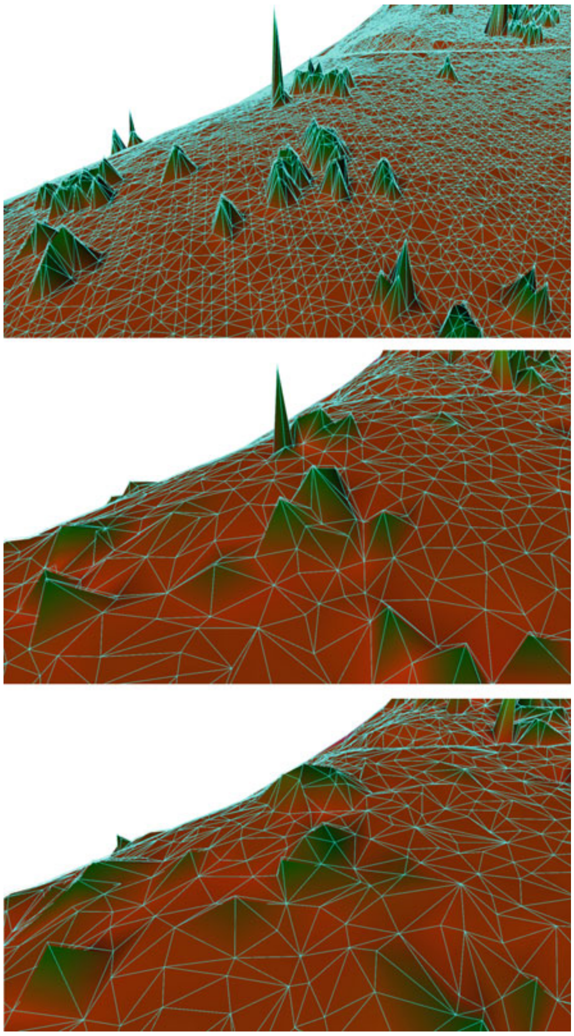
Fig. 8 Preservation of salient features after 8 analysis steps. Top image: high-density, high-detail input (1.6 M faces), middle image (160 K triangles): the combined QEM and thin plate energy cost, bottom image (158 K triangles): the graph coloring strategy used in [27, 37]

3.3 Vertex Frequency Analysis Interpretation

Vertex frequency analysis is a more recently developed subfield of Graph Signal Processing. This subfield discusses solutions for a graph equivalent of time and frequency domain localized transforms such as the windowed fast Fourier transform, the short-time Fourier transform or the windowed fast wavelet transform. Shuman et al. [33] described a solution based on a new definition for graph-based translation operators. Jestrović et al. [23] improved this solution by designing an $O(N^3)$ implementation, as opposed to the $O(N^4)$ complexity of the original. While these developments are sound mathematical generalizations (although some of the

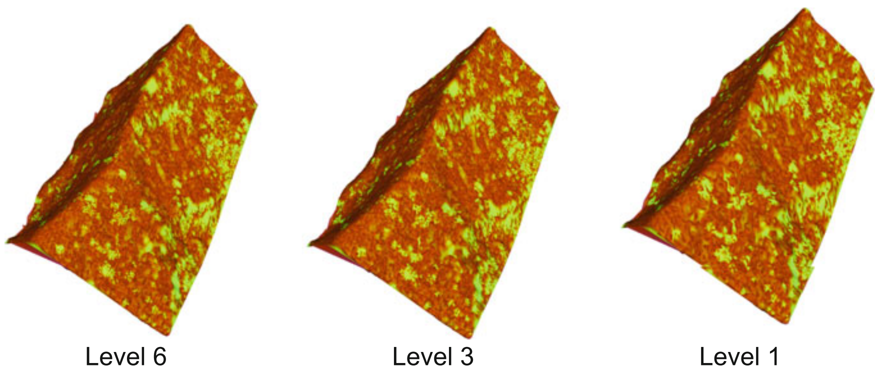

Fig. 9 Kauai Mountain fragment consisting of 79,000 point samples. Represented are level 6, 3 and 1 corresponding to the initial set, the middle resolution and the lowest resolution in a cascade of 5 analysis passes

translation operator properties on graphs are no longer preserved with respect to the one-dimensional case), their computational complexity may not lend them useful for processing very large sets that are common in most applied problems.

Using the wavelet coefficients resulted from a cascade of analysis steps, we can offer an intuitive interpretation that can partially cover the goal of vertex frequency analysis. Given the nature of lifting scheme wavelet constructions, it is not directly possible to discuss spectral features of the signal. However, the detail vectors associated to the odd samples at each scale offer an intuition in this direction. Typical graph vertex frequency analysis algorithms employ an adapted form of the windowed fast Fourier transform or simply restrict the Laplacian eigendecomposition to a neighborhood of a certain size around each vertex. Detail vectors, on the other hand, are directly associated with a single vertex and a single scale. Thus, the information encoded in these entities is, by definition, localized in space and frequency domain. As opposed to classical vertex frequency interpretations, it is not directly possible to examine more than one spectral component for a specific odd vertex. Nevertheless, the magnitude of the wavelet coefficients is a direct indicator of the strength of the local graph signal with respect to the characteristic frequency band of a level of resolution.

We propose an experiment where we subject a terrain fragment of the Kauai Mountain in Hawaii (extracted from the www.opentopography.org portal), to a sequence of five cascading analysis steps (see Fig. 9). The resulting difference vector amplitudes are then plotted in Fig. 10.

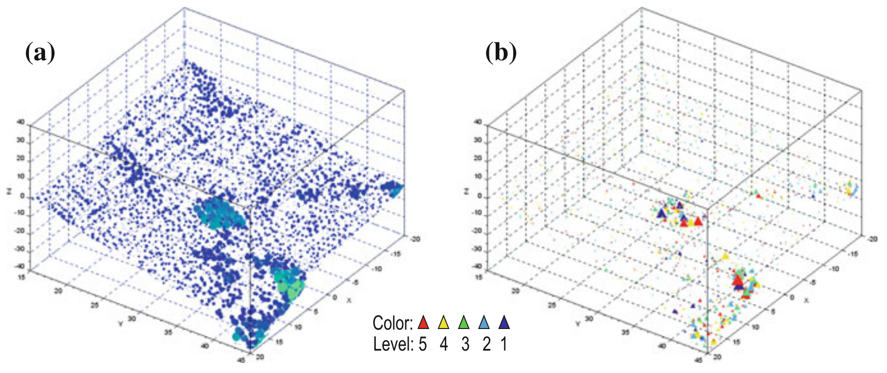

Fig. 10 Difference vector magnitude across 5 consecutive analysis passes of the Kauai fragment Fig. 9. The **a** subplot reveals the height above ground of the initial terrain points, while the **b** subplot contains markers indicating the strength (size) and frequency band of the resulting difference vectors at each of the odd vertices present in the **a** subplot. The regions in **a** with abrupt variations register a high frequency footprint, while similar points clustered together register lower frequency signatures as well

4 Conclusions

Several conclusions can be drawn by analyzing the results obtained using the different lifting designs. First, by using the update designs that are not guaranteed to preserve the signal average for general graphs (e.g. [27]), we observe the RMSE levels have the highest values, regardless of the prediction filter design. As such, the charts Figs. 5a, 6a and 7a reveal that over a progression of 12 analysis steps, the uniform weighing prediction design (UG) is, on average, the better choice. While the geometric Laplacian (GL) and the non-negative least squares weights (NNLS) yield better quality results, after 6 or 7 decomposition steps they become unstable. The use of the cotangent Laplacian weights (CL) is not justified either, since this design takes into account only the geometric structure of the data, regardless of the attribute variability. Overall, both uniform and cotangent Laplacian weights produce more stable results, but the uniform design is to be preferred due to its consistently lower RMSE levels. Next, directly preserving the average of the scale coefficients in a one-ring neighborhood (ORMP) contributes to both stability and error-minimizing properties of the analysis sequence for all sets (charts Figs. 5b, 6b and 7b). The best results are, however, achieved through the use of the minimum norm, mean preserving update weights (proposed by Jansen et al. [22]). This design ensures the highest stability while sensibly decreasing the mean square error. More specifically, for all terrain sets the ORMP design is surpassed by the combined use of Laplacian prediction weights and the minimum norm update vector coefficients. Both of our proposals, the geometric Laplacian (GLMN) and the non-negative least squares (NNLSMN) attain an almost twofold accuracy over the cotangent (CLMN) and uniform weights Laplacian (UGMN). While the RMSE could be reduced even further by lifting the constraints on the least squares prediction filter, as proposed in [27, 37], the scheme becomes numerically unstable due to overfitting of the prediction weights values and the prediction of the boundary vertices. While we can fix the boundary vertices and alleviate the stability issue, our design does not require such vertex selection constraints to be applied.

The multiresolution experiments conducted with the three samples also confirm the 25% reduction ratio of the number of vertices after each downsampling step. More specifically, the average reduction rate attained for the Smoky set (Fig. 5) is 27%, for the high-density Carpathian set (Fig. 6) the average ratio is 28%, and for the low-density Carpathian set (Fig. 7) this average ratio is 27%.

As immediate applications for this graph-wavelet multiresolution framework, we suggest filtering [11] and, as experimentally examined in Fig. 10, we also suggest considering the potential of using the detail vector information to offer an intuitive classification similar to that of vertex frequency analysis.

References

1. H.S. Abdul-Rahman, X. Jane Jiang, P.J. Scott, Freeform surface filtering using the lifting wavelet transform. Precis. Eng. **37**(1):187–202 (2013)
2. C. Alberto Silva, C. Klauberg, A.M. Klein Hentz, A.P. Dalla Corte, U. Ribeiro, V. Liesenberg, Comparing the performance of ground filtering algorithms for terrain modeling in a forest environment using airborne LiDAR data. Floresta e Ambiente **25** 00 (2018)
3. M. Bertram, Wavelet analysis for progressive meshes, in *Proceedings of the 23rd Spring Conference on Computer Graphics, SCCG '07* (ACM, New York, 2007), pp. 161–167
4. G. Beyer, Terrain inclination and curvature from wavelet coefficients. J. Geodesy **76**(9), 557–568 (2003)
5. G. Chen, M. Maggioni, Multiscale geometric wavelets for the analysis of point clouds, in , *2010 44th Annual Conference on Information Sciences and Systems (CISS)* (2010), pp. 1–6
6. H. Choi, R. Baraniuk, Multiscale manifold representation and modeling, in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05)*, vol. 4 (2005), pp. iv/569–iv/572
7. T. Cioaca, B. Dumitrescu, M.-S. Stupariu et al., Heuristic-driven graph wavelet modeling of complex terrain, in *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, ed. by Y. Wang, X. Jiang, D. Zhang, vol. 9443 (SPIE Proceedings, 2015)
8. T. Cioaca, B. Dumitrescu, M.-S. Stupariu, Combined quadric error metric and lifting scheme multivariate model simplification. U.P.B. Sci. Bull. Ser. C (2016)
9. T. Cioaca, B. Dumitrescu, M.-S. Stupariu, Graph-based wavelet representation of multi-variate terrain data. Comput. Graph. Forum **35**(1), 44–58 (2016)
10. T. Cioaca, B. Dumitrescu, M.-S. Stupariu, *Lazy wavelet simplification using scale-dependent dense geometric variability descriptors* (J. Control Eng. Appl, Inf, 2016)
11. T. Cioaca, B. Dumitrescu, M.-S. Stupariu, Riemannian filters for multi-variate mesh signals, in *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017) - Volume 1: GRAPP, Porto, Portugal, February 27–March 1, 2017* (2017), pp. 228–235
12. R.R. Coifman, M. Maggioni, Diffusion wavelets. Appl. Comput. Harmon. Anal. 21(1):53 – 94, 2006. Special Issue: Diffusion Maps and Wavelets
13. L. De Floriani, P. Magillo, *Triangulated Irregular Network* (Springer, New York, 2016), pp. 1–2

14. L. Demaret, N. Dyn, M.S. Floater, A. Iske, Adaptive thinning for terrain modelling and image compression, in *Advances in Multiresolution for Geometric Modelling*, ed. by N.A. Dodgson, M.S. Floater, M.A. Sabin (Springer, Berlin, 2005), pp. 319–338
15. J.S. Evans, A.T. Hudak, A multiscale curvature algorithm for classifying discrete return lidar in forested environments. IEEE Trans. Geosci. Remote Sens. **45**(4), 1029–1038 (2007)
16. K Fujiwara, Eigenvalues of laplacians on a closed riemannian manifold and its nets, in *Proceedings of AMS*, vol. 123 (1995)
17. M. Garland, P.S. Heckbert, Simplifying surfaces with color and texture using quadric error metrics, in *Proceedings of the Conference on Visualization '98, VIS '98* (IEEE Computer Society Press, Los Alamitos, 1998), pp. 263–269
18. I. Guskov, W. Sweldens, P. Schröder, Multiresolution signal processing for meshes, in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99* (ACM Press/Addison-Wesley Publishing Co, New York, 1999), pp. 325–334
19. D.K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory. Appl. Comput. Harmon. Analys. **30**(2), 129–150 (2011)
20. M. Isenburg, Lastools–efficient tools for lidar processing, version 150304 (2015)
21. M. Jansen, Multiscale local polynomial smoothing in a lifted pyramid for non-equispaced data. IEEE Trans. Signal Process. **61**(3), 545–555 (2013)
22. M.H. Jansen, G.P. Nason, B.W. Silverman, *Scattered Data Smoothing by Empirical Bayesian Shrinkage of Second Generation Wavelet Coefficients*, ed. by M. Unser, A. Aldroubi, vol. 4478 (2001)
23. I. Jestrović, J.L. Coyle, E. Sejdić, A fast algorithm for vertex-frequency representations of signals on graphs. Signal Process. **131**, 483–491 (2017)
24. W. Jingsong, K. Amaratunga, Wavelet triangulated irregular networks. Int. J. Geograph. Inf. Sci. **17**(3), 273–289 (2003)
25. M. Kalbermatten, D. Van De Ville, P. Turberg, D. Tuia, S. Joost, Multiscale analysis of geomorphological and geological features in high resolution digital elevation models using the wavelet transform. Geomorphology **138**(1), 352–363 (2012)
26. M. Lounsbery, Multiresolution analysis for surfaces of arbitrary topological type. Ph.D. thesis, Department of Computer Science and Engineering, U. of Washington (1994)
27. E. Martinez-Enriquez, A. Ortega, Lifting transforms on graphs for video coding. Data Compression Conference (DCC) **2011**, 73–82 (2011)
28. M. Meyer, M. Desbrun, P. Schrder, A.H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in *Visualization and Mathematics III. Mathematics and Visualization*, ed. by H.-C. Hege, K. Polthier (Springer, Berlin Heidelberg, 2003), pp. 35–57
29. M. Pauly, M. Gross, L.P. Kobbelt, Efficient simplification of point-sampled surfaces, in *Proceedings of the Conference on Visualization '02, VIS '02* (IEEE Computer Society, Washington, DC, 2002), pp. 163–170
30. F. Payan, M. Antonini, Mean square error approximation for wavelet-based semiregular mesh compression. IEEE Trans. Visual. Comput. Graph. **12**(4), 649–657 (2006). July
31. R. Ronfard, J. Rossignac, Full-range approximation of triangulated polyhedra. Comput. Graph. Forum **15**(3), 67–76 (1996)
32. W.J. Schroeder, J.A. Zarge, W.E. Lorensen, Decimation of triangle meshes. SIGGRAPH Comput. Graph. **26**(2), 65–70 (1992)
33. D.I. Shuman, B. Ricaud, P. Vandergheynst, Vertex-frequency analysis on graphs. Appl. Comput. Harmon. Anal. **40**(2), 260–291 (2016)
34. J.P. Surez, A. Plaza, Four-triangles adaptive algorithms for rtin terrain meshes. Math. Comput. Model. **49**(5), 1012–1020 (2009)
35. W. Sweldens, The lifting scheme: a custom-design construction of biorthogonal wavelets. Appl. Comput. Harmon. Anal. **3**(2), 186–200 (1996)
36. M. van Kreveld, *Digital elevation models and tin algorithms, in Algorithmic Foundations of Geographic Information Systems* (Springer, Berlin, 1997), pp. 37–78

37. R. Wagner, H. Choi, R. Baraniuk, V. Delouille, Distributed wavelet transform for irregular sensor network grids, in *2005 IEEE/SP 13th Workshop on Statistical Signal Processing* (2005), pp. 1196–1201
38. M. Wardetzky, S. Mathur, F. Kälberer, E. Grinspun, Discrete laplace operators: No free lunch, in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07* (Eurographics Association, Aire-la-Ville, Switzerland, 2007), pp. 1196–1201