Level of detail visualization of scalar data sets on irregular surface meshes

Georges-Pierre Bonneau *   Alexandre Gerussi

LMC - CNRS

Abstract

In this article, we build a multi-resolution framework intended to be used for the visualization of continuous piecewise linear functions defined over triangular planar or spherical meshes. In particular, the dataset can be viewed at different level of detail, that's to say as a piecewise linear function defined over any simplification of the base mesh. In his multi-resolution form, the function requires strictly the same volume of data than the original input: It is then possible to go through consecutive levels by the use of so-called detail coefficients, with exact reconstruction if desired. We also show how to choose a decimation sequence that leads to a good compromise between the resulting approximation error and the number of removed vertices. The theoretical tools used here are inspired from wavelet-based techniques and extended in the sense that they can handle non-nested approximation spaces. The reader might also refer to [2], where a similar framework is discussed for piecewise constant functions.

Keywords and phrases: wavelets, non-regular triangulations, compression, visualization.

1 INTRODUCTION

Our aim is to build a sequence of level of detail representations of continuous piecewise linear scalar data sets defined on irregular planar or spherical triangulations. Our approach is based on vertex removal operations, and combines a local decomposition/reconstruction algorithm inspired from wavelet techniques, together with a global greedy algorithm, in order to construct the levels of detail.
Following our previous work dedicated to piecewise constant data sets ([2]), we use a local wavelet-like decomposition, that maps a functional space onto a coarser space. The fact that the coarser space has no to be a subspace of the finer one (which must be the case in wavelet theory) enables to deal with irregular triangulations.
At the global level, the choice of the order of removal of the vertices is made by a greedy algorithm that is guided by a simple error criterion.

In section 2, we briefly explain the global greedy algorithm. Section 3 describes the functional spaces and the basis functions resulting from the vertex removal process. Section 4 is dedicated to the local decomposition/reconstruction at each vertex removal. Eventually section 5 presents numerical and visual results on a planar data set, and on a spherical data set with 1.3M faces.

Previous work.
Previous work on level of detail representations are mostly dedicated to surface simplifications, i.e. they deal with geometric data sets, while we restrict ourselves with scalar data sets ([6, 7, 5]). In other words, we are not trying to approximate a surface, we approximate a scalar function defined on a fixed surface.

Schröder and Sweldens ([8]) have introduced spherical wavelets to build sequences of level of detail approximations to scalar data sets on the sphere. But since their approach relies strictly on the framework of wavelet theory, it is restricted to data sets defined on regular triangulations constructed by recursive 4-split of a base mesh.

Papers on level of detail for TIN (Triangular Irregular Network) deal with scalar data sets (height values) on irregular planar triangulations (see for example [4]). [1] deals with multivariate scalar data sets on arbitrary surface meshes. Like our paper, they are mostly based on vertex removal. But the crucial difference is that, after each vertex removal, we modify the values of the adjacent vertices, while they leave these values unchanged. In section 5, we will give some visual and numerical experiments that compare sub-sampling results with our algorithm, for the same sequence of vertex removal.

2 Global level algorithm

Our algorithm is based on vertex removal operations. At the global level, a greedy algorithm based on a simple cost function is used to decide the order in which the vertices are removed. If M is a vertex with scalar value α, and M_i are the vertices adjacent to M, with value α_i, we use the following cost for the removal of vertex M:

$$\text{cost of removal} = \left| \frac{\sum_i 1 + \text{distmax} - d(M, M_i)\alpha_i}{\sum_j 1 + \text{distmax} - d(M, M_j)} - \alpha \right|, \quad (1)$$

where $d(M, M_i)$ is the geodesic distance between M and M_i, and distmax $= \max_i d(M, M_i)$.

We could have used the exact L_2 error induced by the removal of each vertex, since our local decomposition is based on L_2 approximation. But we found that the simple cost function (1) reduces the computation time noticeably and still works well in practice.

In addition to the cost function (1), it is possible to forbid the removal of some vertices in order to preserve important features in the data set. We have used this technique to preserve coast-lines in the earth data set (see section 5).

*LMC-CNRS, BP.53, F-38041 Grenoble Cedex 9, France.

3 Functional spaces and basis functions

As stated in the introduction, we want to deal with piecewise linear functions. In fact in this paper we'll show results for such functions in two cases: a planar triangular mesh and a spherical one. The spherical case can be treated almost as the planar one, except for some details (for example the distances on the sphere are the geodesic distances, the re-triangulation criteria is based on a convex-hull property). We will now implicitly consider, in our illustrations, planar meshes.

We start with a 2D-triangulation T_n of n vertices $M_1, \ldots, M_n$. The global greedy algorithm described in section 2 is used to compute a sequence $T_n, T_{n-1}, \ldots, T_m$ of simplified triangulations. The domain covered by T_i will be denoted by $D(T_i)$.

By removing a vertex M from the mesh, the triangulation is only modified locally, over the so-called polygon of influence (PI) of M, which is the polygon whose vertices are adjacent to M (the 1-neighbours of M). This is illustrated in figure 1.

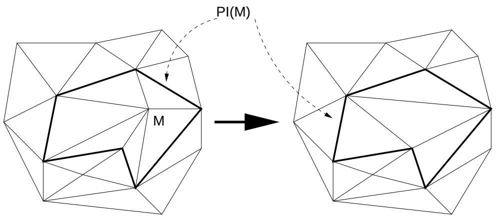

Figure 1: removing of a vertex from a triangulation

There are several ways to choose the new edges which re-triangulate the polygon. Our current implementation is based on a local greedy algorithm in a first step, followed by a local swap of the diagonals in each quadrilateral, in order to ensure the convexity of the triangulation for spherical triangulations, or an empty circle property for planar triangulations.

The sequence T_i, $i = m, \ldots, n$ is used to define the functional spaces forming our multi-resolution analysis: Let V_i, $i = m, \ldots, n$, be the space of all functions that are defined and continuous over $D(T_i)$ and piecewise linear over each triangle of the mesh T_i (shortly: a CPLF, for continuous piecewise linear function).
The definition of a function of V_i is equivalent to the knowledge of its values at every vertex of T_i, and consequently the dimension of V_i is i. The basis of V_i is given by the i hat-functions: the hat-function of the vertex M_j in V_i takes value 1 at M_j and then declines linearly to zero on each triangle of the PI of M in T_i (see figure 2 for an illustration).

4 Local decomposition/reconstruction

The problem of computing the decomposition/reconstruction is similar at each level. Consequently, we shall now focus on the transition between V_i and V_{i-1} for a fixed i, or, to hide the index i, between V^f and V^c (f=fine, c=coarse), where V^c results from the removal of vertex M in V^f.

Our decomposition/reconstruction process is based on L_2-approximation between V^f and V^c. More precisely, given a function $F^f \in V^f$, we want to compute two things:

- its L_2-approximation F^c in V^c (which is the orthogonal projection of F^f in V^c),

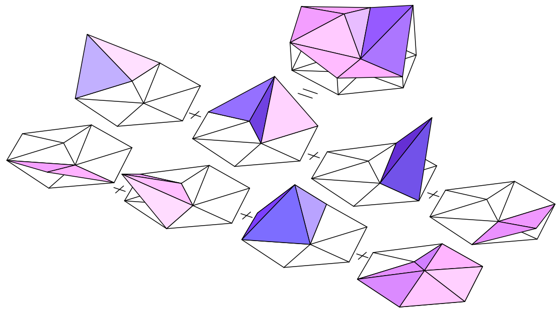

Figure 2: Example of a CPLF and its decomposition in the hat function basis

- a detail coefficient b, encoding the loss of information due to projection, and allowing exact reconstruction during the synthesis process.

This decomposition/reconstruction process is inspired from wavelet-based techniques, and extents them, in the sense that it can be applied even if the coarse space V^c is not included in the fine space V^f. This extension has been introduced in a previous work in a different context ([3], section 2). The reader might refer to [2] or [3] for a context-independent presentation of this extension.

4.1 Basis functions in V^f and V^c

Before going further with the actual computations, we should have a careful look at the basis functions in V^f and V^c. The hat function corresponding to the vertex M no longer exists in V^c. This is the reason why

$$\dim(V^f) = \dim(V^c) + 1.$$

Also, the hat functions based on the neighbours of M in T^f disappear, since the vertex removal creates a new neighbourhood for all of them, but they are "replaced" by hat functions defined on the same vertices and their new neighbourhoods in T^c. All the other basis function remain unchanged. By comparing the supports of the basis functions in figure 3, you can see the consequences of a vertex removal on the basis functions whose support overlaps the PI of the removed vertex.

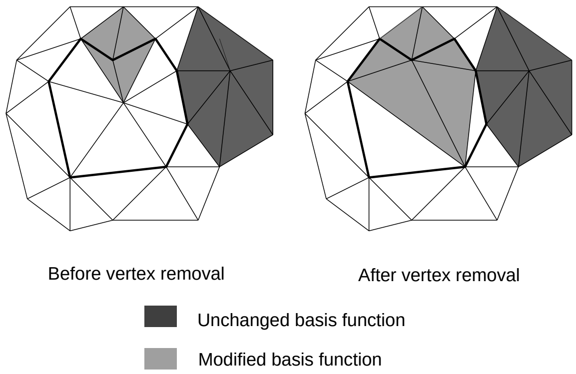

Figure 3: Supports of hat basis functions, before and after vertex removal.

We note M_i^f, $i = 0, \ldots, k$ and M_i^c, $i = 1.., k$ the basis of V^f and V^c, respectively. M_i^f is the hat function centered on vertex M_i in T^f. We have just seen that, in fact, if we number 1 to l the 1-neighbours of M (in T^f) then

$$M_i^f = M_i^c \quad \forall i = l+1, \ldots, k.$$

To perform L_2-approximation between V^f and V^c, we will have to compute the following matrices of scalar products between finer and coarser basis functions:

$$\begin{array}{ll} G^f & = (< M_i^f, M_j^f >)_{i=0,\dots,k,\ j=0,\dots,k} \\ G^c & = (< M_i^c, M_j^c >)_{i=1,\dots,k,\ j=1,\dots,k} \\ U^{cf} & = (< M_i^c, M_j^f >)_{i=1,\dots,k,\ j=0,\dots,k} \\ A & = (G^c)^{-1}\, U^{cf} \end{array}$$

G^c and G^f are the Gram-schmidt matrices in V^c and V^f respectively. And for those who are familiar with the terminology of wavelet theory, A can be seen as the matrix of scalar products between the fine scaling functions (in V^f) and the dual coarse scaling functions (in V^c).

4.2 Decomposition

The decomposition step takes as input a function $F^f = \sum_{i=0}^{k} a_i^f M_i^f$ in the finer space V^f, and outputs an approximation $F^c = \sum_{i=0}^{k} a_i^c M_i^c$ in V^c and a detail coefficient b. Note that there is only one detail coefficient since $\dim(V^f) = \dim(V^c) + 1$. This detail coefficient is conceptually linked to the vertex M, it can be stored in place of a_0^f.

Coarse coefficients
The coarse coefficients $(a_.^c)$ are computed from the fine coefficients $(a_.^f)$ with the following formula:

$$(a_.^c) = A(a_.^f)\,.$$

In fact the matrix A ensures that F^c is the best L_2-approximation of F^f in V^c.

Detail coefficient
The detail coefficient is computed from the $(a_.^f)$ coefficients through the equation:

$$b = B(a_.^f),$$

where B is a line-vector representing a dual wavelet function, in the basis of V^f (see [2] or [3]).
This vector is computed as follows: First we know that it has to fulfill the orthogonal property $A\left(G^f\right)^{-1} B^T = 0$ (which means in analogy with wavelet theory, that the dual wavelet is orthogonal to the dual coarse scaling functions). This gives a one-dimensional space of solutions for the line-vector B. B is then determined uniquely by norming it to 1 in V^f: $B\left(G^f\right)^{-1} B^T = 1$. The orthogonalization and normalization on B ensure that the detail coefficient b is a measure of the approximation error between the input function F^f and the output function F^c.

4.3 Localization

One problem of importance is that the matrices G^c, G^f and U^{cf} have to be computed at each vertex removal, and this is a too big amount of work since these matrices, although sparse, can potentially be very large. Nevertheless, since a vertex removal only changes basis functions in a small neighbourhood, it is quite intuitive that the best approximation of F^f is not going to differ very much from F^f outside of this neighbourhood.

Precisely, we parameter our decomposition/reconstruction process with a integer $K \geq 1$ that reduces the global problem to a local one by doing the following: Instead of working with all basis functions, we only consider the basis functions which are based on vertices which are $i-$neighbours of M in T^f, $i = 1, \dots, K$ (see figure 4). This means that F^c is no longer the best L^2-approximation of F^f but it is supposed to be very close to. We won't try to quantify this difference here, but this assumption led to good results ($K = 2$ or 3 appeared to be large enough).

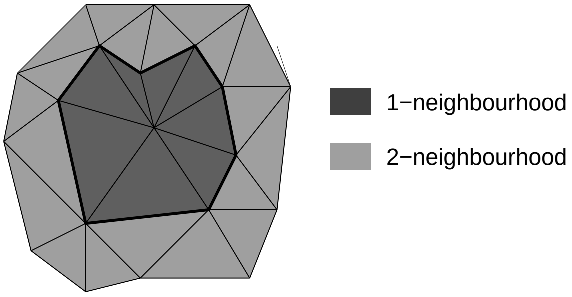

Figure 4: the 1- and 2-neighbourhood of a vertex

Another consequence of this strategy is that the global algorithm is now in $o(n)$, because each local analysis/synthesis is made in constant time, independently of the number of vertices in T_i.

4.4 Computing the scalar products

Let's say a few words about the computations of the matrices G^c, G^f and U^{cf}. Their coefficients are scalar products between the functions $M_.^f$ and $M_.^c$. Because $M_i^f = M_i^c$ whenever $i > l$, these three matrices have a lot of coefficients in common. Moreover, a function only has non zero scalar product with its 1-neighbour functions, and the support of two such functions only overlap over two triangles (see figure 5).

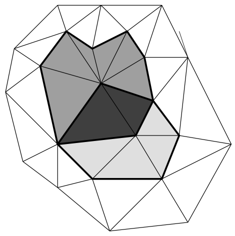

Figure 5: intersection of the supports of two basis functions.

In fact, the only difficult cases occur in the computation of scalar products of mixed type $< M_i^f, M_j^c >$ when $i \leq l$ and $j \leq l$, because it is then necessary to compute the intersections between the fine and coarse triangles forming the inside of the PI of M in T^f and T^c respectively. Nevertheless, without sinking into too much detail, this problem is simpler than the general problem of finding the intersections between any two triangle sets, because the PI is a star-polygon, centered on the removed vertex.

Finally, the computation of the matrices only requires a reasonable amount of work, and this is crucial because our framework is intended to run on possibly very large datasets (see the next section and the application on the 1.3M faces spherical data set).

4.5 Reconstruction

The reconstruction process is relatively straightforward: the synthesis matrix is nothing but the inverse of the analysis matrix, thus, the fine coefficients $(a_\cdot^f)$ can be recovered from the coarse coefficients $(a_\cdot^c)$ and the detail coefficient b with the following formula:

$$(a_\cdot^f) = \begin{pmatrix} A \\ B \end{pmatrix}^{-1} \begin{pmatrix} a_\cdot^c \\ b \end{pmatrix}$$

4.6 Illustration

We shall conclude section 4 with an illustration of our decomposition algorithm. Figure 6 shows at the top left an input function (the removed vertex is circled), at the top right the output function, and the bottom shows two views of the corresponding wavelet function. This decomposition was computed using the localization parameter $K = 2$.

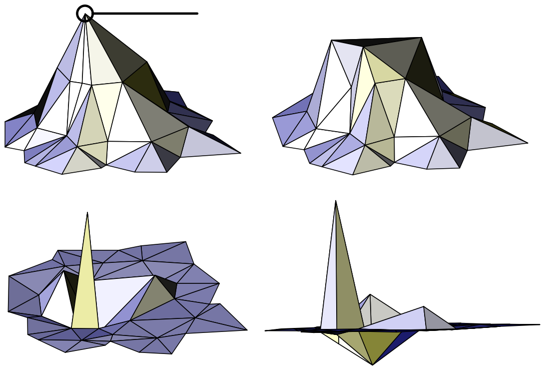

Figure 6: Approximating a CPLF: top left = input function, top right = approximation, bottom = wavelet function (two views). The removed vertex is circled.

5 Results

We start this section with some numerical examples of decomposition/reconstruction on a data set that was specifically constructed to test the stability of the algorithms. We have chosen a strongly irregular planar triangulation, and have mapped a non-smooth image on that triangulation: the triangulation was generated by Delaunay-inserting random points in a square domain, and the data value at each vertex was sampled from the ETOPO5 data set from the National Oceanographic and Atmospheric Administration [1], showing the elevation/bathymetry of the earth.

The data set has 25000 vertices. The coast-lines were preserved by the global greedy algorithm, as explained in section 2: the vertices to keep were detected by looking if a change of sign occured at the 1-neighbour vertices. The decomposition was computed using the localization parameter $K = 2$ (see section 4.3). We compare the results of our decomposition algorithm with the results obtained by simply sub-sampling the data set. Figure 7 shows the relative L^2-error versus number of vertices for our decomposition, and for sub-sampling. Figure 8 shows two partial reconstructions with 2500 vertices out of 25000, one based on our algorithm (bottom right), and the other based on sub-sampling (bottom left), for the same sequence of removed vertices.

Color plate 1 illustrates our algorithm on a spherical data set consisting of 1.3M faces. The original data set is defined on a regular triangulation (4-to-1 split with 8 levels of subdivision starting on a icosahedron). The data value at each vertex was sampled from the ETOPO5 data set (which consists of 2160 x 4320 samples on a uniform grid).
Our algorithm was applied using the localization parameter $K = 2$ (see section 4.3), and with preservation of the coast-lines.
The upper part of color plate 1 compares the result our algorithm, with the result obtained from sub-sampling the data set: (a) shows the approximation resulting from our algorithm, and (c) shows the sub-sampling result. (b) shows the corresponding spherical mesh.
The bottom part of color plate 1 shows different partial reconstructions computed from our algorithm: (g), (h), (i) show the spherical meshes corresponding to (d) (100000 vertices), (e) (200000 vertices) and (f) (300000 vertices).

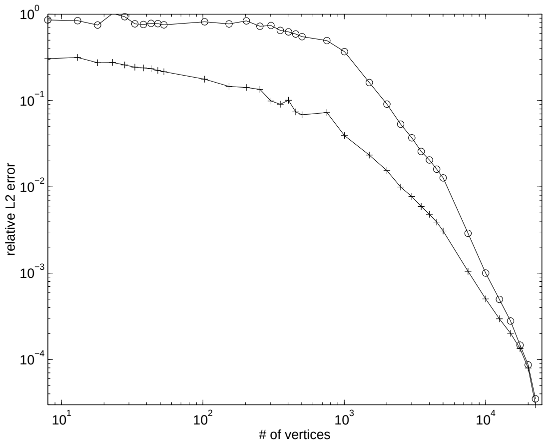

Figure 7: Relative L_2-error vs. # of vertices. 'o':sub-sampling,'+':L_2-approximation

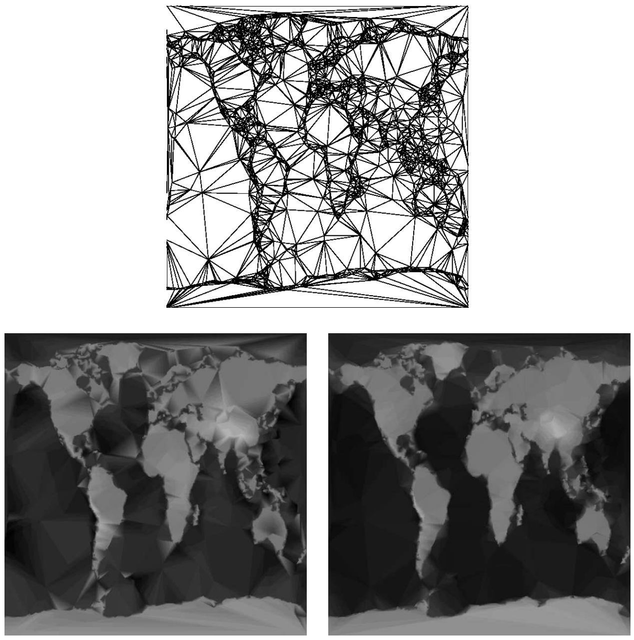

Figure 8: 2500 vertices out of 25000, top=mesh, bottom left=sub-sampling, bottom right=L_2-approximation

[1] Available via anonymous ftp at `ftp://ftp.ngdc.noaa.gov/Solid_Earth/Topography/tbase_5min/`

References

[1] Chandrajit L. Bajaj and Daniel R. Schikore. Error-bounded reduction of triangle meshes with multivariate data. In *Visual Data Exploration and Analysis III*. SPIE, 1996.

[2] Georges-Pierre Bonneau and Alexandre Gerussi. Hierarchical decomposition of datasets on irregular surface meshes. In *Proceedings CGI '98*, 1998.

[3] Georges-Pierre Bonneau, Stefanie Hahmann, and Gregory M. Nielson. BLaC-Wavelets: A multiresolution analysis with non-nested spaces. In *IEEE Visualization '96*, pages 43–48. IEEE, October 1996.

[4] Leila De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Applications*, 9(2):67–78, March 1989.

[5] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[6] Hugues Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 99–108. ACM SIGGRAPH, August 1996.

[7] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH 97 Conference Proceedings*, pages 189–198. ACM SIGGRAPH, August 1997.

[8] Peter Schröder and Wim Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *SIGGRAPH 95 Conference Proceedings*, pages 161–172. ACM SIGGRAPH, August 1995.