Accurate Triangular Regular Network adjustment to Large Digital Elevation Models

José M. Santana
University of Las Palmas de G.C.
Imaging Technology Center (CTIM)
Spain (35017), Las Palmas de G.C.
josemiguel.santana@ulpgc.es

Agustín Trujillo
University of Las Palmas de G.C.
Imaging Technology Center (CTIM)
Spain (35017), Las Palmas de G.C.
agustin.trujillo@ulpgc.es

José P. Suárez
University of Las Palmas de G.C.
Division of Mathematics, Graphics and Computation (MAGiC), IUMA
Spain (35017), Las Palmas de G.C.
josepablo.suarez@ulpgc.es

Sebastián Ortega
University of Las Palmas de G.C.
Division of Mathematics, Graphics and Computation (MAGiC), IUMA
Spain (35017), Las Palmas de G.C.
sebastian.ortegatrujillo@gmail.com

ABSTRACT

Nowadays, large volumes of terrain data are available to use as *Digital Elevation Models*, from which coarser meshes can be progressively generated for visualization and other purposes. Previous studies compared different methods to adjust those meshes, concluding that no method performs the best for all kinds of terrain. In this work, a pipeline to accurately adjust TRNs to DEM is proposed. An initial approximation is calculated by solving a linear system from input data. Vertices with a major contribution to the global error of the mesh are then tuned using a local refinement algorithm. Experimentation shows that meshes adjusted using the proposed pipeline fit better the original DEM than ones generated using classic methods as linear interpolation for several benchmark elevation models.

Keywords
Triangular Regular Networks, adjustment, Digital Elevation Model, level of detail, terrain visualization.

1 INTRODUCTION

The growing capacity to adquire and store data has allowed that big volumes of terrain data are currently available for the general public, in which the earth is modelled with fine resolutions. These data are presented as *Digital Elevation Models* or DEM, which can be structured in several raster and vector formats. The visualization of large terrain models in applications like virtual globes [Tru12a] requires an intensive use of computation and memory, so it is usual to generate multi-resolution schemes [Lue02a] in which different, progressively coarser meshes represent points in the original model. In this regard, the number of triangles that a 3D scene is going to display is highly dependent on the implemented LoD test, which ideally limits the graphics requirements for any given point of view [Sua15a].

We are particularly interested in finding a method that maximizes the vertical accuracy of the generated mesh in relation with the model. In this regard, some studies have researched on how to generate surface points from discrete elevation models. These techniques are ultimately used to generate continuous terrain representations that try to fit the original model as well as possible. Comparatives between different interpolation methods have been introduced in [Agu05a] and [Cha06a], from which can be extracted that no method performs the best for all terrain scenarios, being more convenient to use different methods depending on the terrain to be represented. It is also worth considering that, for multi-resolution schemes and visualization purposes, vertical accuracy between two inmediate levels of detail should also be maximized in order to avoid popping artifacts in dynamic multi-lod visualizations whenever possible.

In this study, a pipeline to adjust TRNs to fit large DEMs is proposed. An initial mesh is obtained by approximating the solution of a linear system from the input data. After that, a refinement method tunes vertices to optimize the global error of the mesh. Finally, experimentation has been conducted to ensure the applicability of our proposed algorithm to very large DEMs. In this regard, a tiling strategy to speed up the process is analyzed. Tests are also conducted to check the behavior of the whole pipeline, compared with classic methods, in a benchmark elevation model.

1.1 Related work

There are several methods to generate meshes from terrain data. One of the most popular approaches is the use of *Triangular Irregular Networks* (TIN) [Peu78a], which minimize the amount of triangles to be rendered given an acceptable error threshold. Different authors, such as Hoppe [Hop98a], Puppo [Pup96a] or El-Sana and Varshney [Els99a], have worked with TIN applied to multi-resolution schemes and variable mesh resolutions. Other option is the use of regular grids and *Triangulated Regular Networks* (TRN) [Agu03a], which are easier to implement and consists of triangle meshes that represent a surface in which all vertices are placed on a regular grid. In this line, the studies of Lindstrom et al. [Lin96a] and Pajarola [Paj98a] can be highlighted. More approaches on TRN and TIN are introduced in a survey from Pajarola and Gobbetti [Paj07a].

An alternative approach, known as Right-Triangulated Irregular (RTIN) Networks, has also been discussed in the literature. This variation of the TIN structure places the vertices on a grid, but still uses fewer triangles where they are not needed. However, when presented as raster information to the GPU, this scheme is not more efficient in terms of memory, and saving triangle drawing implies a local lost of detail. The work of Suárez and Plaza [Sua03a] describes a pipeline to refine this kind of model.

At the present time, lines of work are more focused on taking advantage of the capacities of GPU for terrain visualization. Authors such as Yusov and Shevtsov [Yus11a] or Kang et al. [Kan15a] have proposed to perform the triangulation task in the tessellation stage of OpenGL, saving computation time and data transfers between CPU and GPU. However, on-the-fly tessellation of surfaces is an intensive process that is not available on many graphics pipelines, e.g. mobile platforms implementing OpenGL ES 2.0. Besides, the downscaling of these models normally relies on linear interpolations of the underlying DEM, which does not assure a solution with the best height accuracy.

2 DEM REPRESENTATION ERROR METRIC

The proposed system utilizes a linear system to approximate a terrain sector using TRNs of resolutions lower than the one of the provided DEM. Given the input data defining punctual values of the terrain elevation, the proposed method adjusts the TRN by approximating the solution of a linear system. For any point P_l of the input elevation data, it is determined which triangle of the final triangular mesh is going to represent that particular point in cartographic space. As the triangles do not overlap on the latitude/longitude plane, this calculation is rapidly achieved by mapping the geographic coordinates of P_l to the barycentric coordinates of the tested triangle.

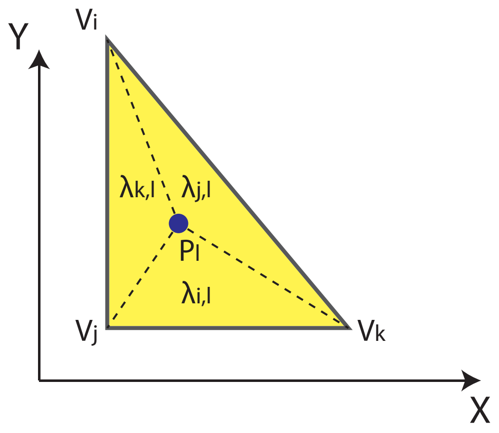


Figure 1: Barycentric coordinates of P_l

Thus, for all triangles present on the final TRN, the algorithm defines the projection of its vertices (V_i, V_j, V_k) in geographic space, by linearly interpolating their position within the TRN geographic sector. On this 2D triangle, the coordinates of P_l are transformed to its barycentric coordinates as depicted in Figure 1. If the three barycentric components are equal or greater than zero, the projection of P_l falls into the triangle area.

Barycentric coordinates are homogenous so they can be used to interpolate any point within the triangle. By using normalised barycentric coordinates, any inner point can be computed as a weighted sum of the triangle vertices. This coordinate system conversion can be efficiently achieved applying Cramer's rule.

Considering an input elevation data of n points and a desired TRN of t triangles, the limiting behavior of the algorithm that composes the linear system is of $O(n*t)$ asymptotically. This could lead to efficiency problems due to these magnitudes growing exponentially as both the DEM and the TRN increase their geographical space or spatial resolution.

In practice, however, the regularity of TRNs can be exploited to speed up the generation of the barycentric coordinates. First, all the DEM points must be expressed as x, y, z coordinates, orthogonal to the grid axis. Then, the normalized x, y coordinates, relative to their grid cell inner coordinates, can be expressed as:

$$\{x_C, y_C\} = \{\frac{((x - X_0) \bmod \Delta X)}{\Delta X}, \frac{((y - Y_0) \bmod \Delta Y)}{\Delta Y}\} \quad (1)$$

where $\{X_0, Y_0\}$ are the origin of the TRN, and $\{\Delta X, \Delta Y\}$ are the distances in X and Y between two consecutive vertices. The barycentric coordinates can then be computed in linear time, within a set of 2 two triangles. To determine in which triangle each point falls, the condition $x_C > y_C$ indicates a point belonging to the upper triangle.

For each point in the input data, its corresponding projection on the TRN can be calculated as an interpolation of the three vertices of the triangle weighted by the barycentric coordinates calculated on the previous step. On that assumption, the error function for a particular point P_l on the input set is given by Equation 2:

$$E_l = P_{l_z} - (\lambda_{i_l} V_{i_z} + \lambda_{j_l} V_{j_z} + \lambda_{k_l} V_{k_z}) \quad (2)$$

In this equation, i, j and k are, for the point P_l, the indices of the vertices of its enclosing triangle, and λ_{i_l} are the barycentric coordinates of $P_{\text{l xy}}$ on the $V_{\text{i xy}}$, $V_{\text{j xy}}$, $V_{\text{k xy}}$ projection of the triangle V_i,V_j,V_k with respect to the V_i vertex.

Given a similar expression for every point of the DEM, we can express the resulting linear system as a matrix equation of the form $A \cdot X = B$.

The A matrix is a large non-square sparse matrix which contains the coefficients of Equation 2. These coefficients are the above-mentioned barycentric coordinates computed in 2D. This way, for any given row l, it expresses the error function of the input datum P_l.

The B vector is conformed by $P_{\text{l z}}$, which are the heights associated to the P_l coordinates based on the input data. Finally, the X vector represents the to-be-computed heights of all the vertices of the TRN.

This linear system is conformed by a coefficient matrix of size $n \times (q \cdot w)$, being n the number of input points on the DEM and $(q \cdot w)$ the resolution of the desired TRN. This final size can be extremely high considering the size of most available DEMs.

The solution of this system, in case of existing, is a set of heights from which a triangulation that perfectly matches the provided DEM can be generated. For most cases, the system is unsolvable and the goal is to find an X that optimises the vertical accuracy by minimising $\mu(\|A \cdot X - B\|)$. Ultimately, $\mu(A \cdot X - B)$ is the mean error produced by the method for a given TRN resolution.

3 METHOD PROPOSAL

There are many ways to approximate the solution of the above-mentioned linear system. The Least Square Method [Pai82a] is a regression analysis technique that, given the equations system, minimises the value $\|(A \cdot X - B)\|$ by exploring the space of solutions via gradient descent. This numerical approach is a suitable option for our case, as it does not require the matrix to be square, and is specially efficient for sparse matrices like A. Finally, the execution of the Least Square Method can be accelerated by providing a X_0 from which the space of solutions starts to be explored. In our case, we have calculated this first approximation to the solution by assigning to every TRN vertex an initial height equivalent to the closest point on the DEM.

In case many levels of detail are computed for the same DEM, a multi-grid strategy can be followed. This technique is broadly used to approximate solutions of linear systems that represent spatial characteristics at different resolutions [Qua16a]. In our case, the TRNs can be generated from the finest to the coarsest. Thus, an alternative value for X_0 could be provided by the immediately coarser model, improving the initial approximation and shortening the computation time.

After obtaining a first approximation to the solution, we propose to use an iterative stage to tune vertices with a strong contribution to the global error. Every iteration in the algorithm processes the input solution as follows:

1. A list of the contribution to the overall error of each one of the vertices of the TRN, Q, is obtained by the following expression : $Q = A^T \cdot E$, where A is the matrix presented in Section 2 and E is the result of the expression $A \cdot X - B$.
2. For all vertices in the TRN whose Q value is non-zero, a new value for the vertex is computed using the expression $V_\text{n} := V_\text{n} - \alpha \cdot Q_\text{n}$. This will be the height of the vertex in the following iteration only if the change decreases the global error. Otherwise, the vertex remains at the same height. In the rule, α is a factor which ranges between 0 and 1. Experimentally, a value of 0.7 gives us good results.
3. The process stops when the error difference between two iteration is below a tolerance of 10^{-10} meters.

4 LARGE DEM SIMPLIFICATION PIPELINE

The main drawback of the iterative stage proposed in the previous section is its execution time for large TRNs. A higher number of iterations could be needed to find the solution, each of them modifying vertex heights and checking the error for a rising number of vertices in the TRN, thus, heavily increasing the execution time.

A straightforward technique we propose to reduce the complexity of the linear system is to subdivide the desired TRN into tiles that cover a maximum number of

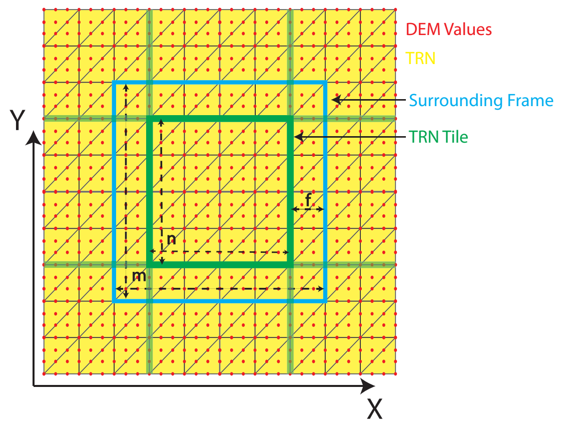


Figure 2: Division in tiles of a TRN

data points. Considering a regular axis-aligned DEM, a n-fold subdivision in each dimension of the DEM sector generates n^2 linear systems, reducing their coefficient matrix by a factor of $1/n^4$. Thus, the overall dimensionality of the systems to be solved is reduced by a factor of n^2 by tiling. Furthermore, the generated linear systems are independently solvable, hence it is easier for the available hardware to parallelise their resolution, taking advantage of multi-core and distributed architectures. However, the TRN-DEM adjustment is not independent between different zones, and tiling induces a penalty on the error results, since border vertices lack information about neighboring areas outside of the tile, leading in an inaccurate adjustment. To avoid undesired effects, a *frame area* should be added so the vertices of the tile do not present significant variations against the same vertices when the entire mesh is computed at once. The results for that frame are dismissed, using only the results for the tile itself. Such a configuration can be seen in Figure 2.

When the used frame area has enough width, the height values for the boundary of two adjacent tiles should not have appreciable differences and thus it is possible to treat tiles independently. However, as a safety mechanism, the average of the values of a boundary vertex in both tiles is taken to address any discontinuity that could happen during the process. Experimentation has been conducted to determine the ideal width of the referred frame and can be read in Section 5.2.

5 EXPERIMENTATION

During this work, experiments have been designed to test the impact of the proposed splitting strategy, find the best size for a tile and check the accuracy and goodness of the proposed solution. The following subsections present detailed descriptions of each experiment and its results.

5.1 Case of study and machine specs

Two different elevation models have been used during the experimentation work. The first elevation model

	Tenerife	**Puget Sound**
Resolution	2089 x 2849	7169 x 6657
Minimum height	0	0
Maximum height	3689	2429.6
Mean	326.05	230.38
Standard deviation	604.72	355.31
Maximum cliff	504	484

Table 1: Resolution and characteristics of the chosen datasets. All height data are expressed in meters.

covers the Puget Sound region (NW coast of the United States), is downloadable at the website [Pug00a] and has been used as a showcase of DEM analyses by other works such as [Kan15a],[Los04a] or [Liv09a]. The second elevation model covers the island of Tenerife. It was chosen due to its variable morphology and it can be downloaded from the NASA Worldwind services, specifying a bounding box ranging from 27.99°to 28.59°in latitude and -16.9322°to -16.0978°in longitude. Table 1 contains some information about these elevation models. There, *maximum cliff* refers to the maximum difference between a point and its neighbours.

All experiments in this section have been executed using Matlab R2013a and run in a machine whose internal specifications include a CPU Intel Core i7-4790 at 3.60 GHz with a main memory size of 8 GB and uses Windows 8 as a operating system.

5.2 Determining whether splitting is possible for a large TRN

This first experiment checks the error penalty produced due to splitting a TRN in tiles, and particularly, how errors induced by splitting are distributed in the tile. We are checking if differences over a maximum allowed error are found near the edges of the tile after comparing the results obtained from calculating a tile and computing the whole TRN. For the sake of this experiment, that tolerance was set to 0.1 m. If this hypothesis can be confirmed, splitting in tiles can be considered and a frame margin of f vertices of width will be set and added to every tile area to be calculated so the error-prone border area can be discarded.

Let us consider *ratio* as the quotient between the length of the edge of a squared DEM and the length of the edge of a squared TRN to be generated from it. For different chosen ratios, different squared tiles of $n \times n$ vertices are selected, representing $N \times N$-point areas of the DEM. Using the pipeline, the tiles are adjusted independently so they fit their corresponding areas in the DEM. A TRN for the whole DEM is also adjusted at once. After the adjustment, differences between the tile mesh and the tile area in the global TRN have been measured. The relation between the error induced on a TRN vertex and its distance to the border of the tiles

Ratio	*n*	*N*	*f*
1:2	49	97	5
1:2	241	481	5
1:2	497	993	6
1:4	49	193	7
1:4	241	961	10
1:4	497	1985	8
1:8	49	385	7
1:8	241	1921	7
1:8	497	3969	9

Table 2: Frame margin f needed for different tile sizes and levels of detail.

has been analyzed, as it is shown in Figure 3. The pairs of selected n and N values, among with the results of the described analysis are presented in Table 2.

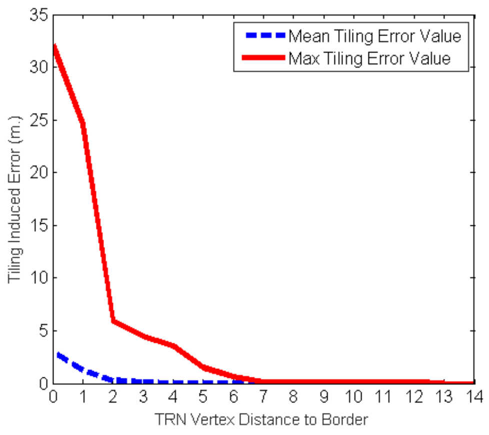


Figure 3: Distribution of differences between meshes for case $n = 241$, $N = 961$ (Ratio = 1 : 4).

As expected, the results show that all errors greater than the tolerance are present in the edges for all the experiments, with a maximum depth of 10 vertices. That leads us to infer it is possible to split the calculation of a TRN in tiles. Adding a frame of width $f = 10$ vertices guarantees that the obtained results will be similar to those achieved by processing the whole DEM at once.

5.3 Minimising execution times for large TRNs

The second test is aimed to determine the proper tile size t that minimises the time required to generate a full large TRN using the proposed pipeline.

For the sake of this experiment, we are trying to generate a TRN of size 3585×3329 from the Puget Sound DEM. For it, a set of different tile sizes are selected so the entire TRN can be split in an exact number of adjacent tiles. We are adding the frame margin f calculated in Section 5.2 to all tiles before processing them using the pipeline. Execution times have been calculated by averaging several execution samples on different DEM areas. Finally, the margin area is discarded, using only the inner area to compound the final TRN, from which a total execution time is also obtained. We will choose the t which minimizes this final execution time. The selected tile sizes and their average execution times can be seen in Table 3.

Tiles	Vertices per tile	Time per tile
8x8	449 x 417	3718
12x12	299 x 278	389.6
16x16	225 x 209	171.7
20x20	180 x 139	130.7
24x24	150 x 119	80.60
28x28	129 x 119	51.43
32x32	113 x 105	40.03
40x40	90 x 84	32.77
48x48	75 x 70	20.97
56x56	65 x 60	15.21
64x64	57 x 53	12.55
128x128	29 x 27	5.295

Table 3: Chosen tile sizes and execution times. All times are expressed in seconds.

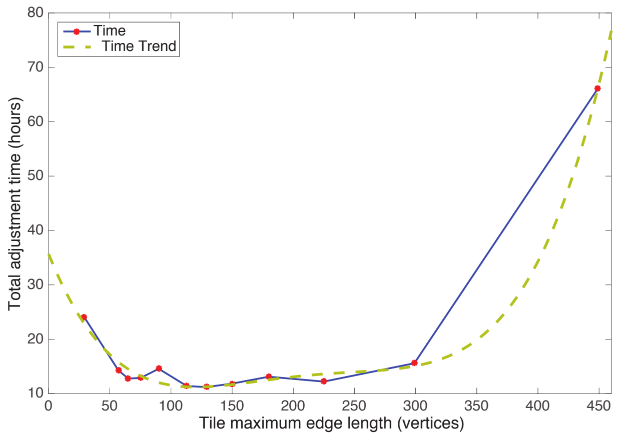


Figure 4: Evolution of estimated time to complete a TRN for the first LoD of the Puget Sound elevation model.

By analyzing the results it is observed that, for large resolutions, the cost of computing a tile is greater than the cost of splitting the tile in four equally-sized subtiles and computing all of them. This can be seen in Table 3: a tile with a resolution of 225×209 needs 171.7 seconds to be computed, whilst the computation of 4 tiles of resolution 57×53 lasts in average 50.2 seconds. On the other hand, very low resolutions have an overhead due to the redundant calculation of vertices which fall in frames of different tiles. The ideal size for the tile should have a cost similar to the one resulting of the computation of its four possible children.

As seen in Figure 4, the ideal t for this experiment is of 113×105 vertices, plus a frame of width f = 10.

5.4 Accuracy of the proposed method

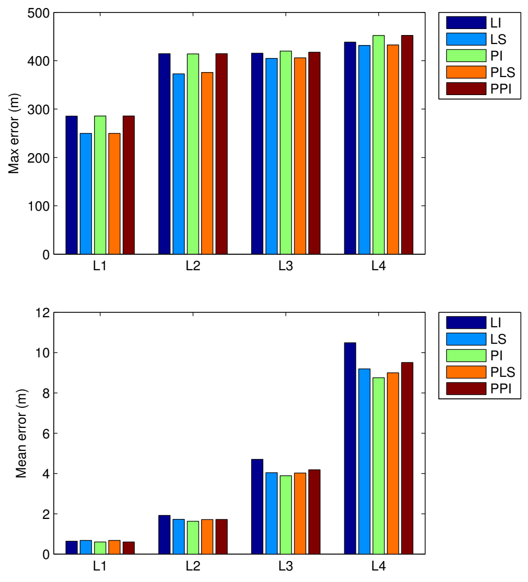


Figure 5: Maximum and mean errors between meshes for levels of detail L1 (most detailed) to L4 (coarsest) and the Puget Sound DEM, for all considered techniques. All units are expressed in meters.

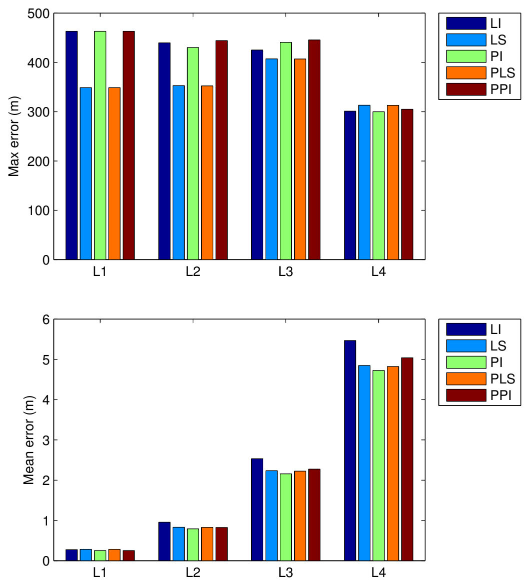


Figure 6: Maximum and mean errors between meshes for levels of detail L1 (most detailed) to L4 (coarsest) and the Tenerife DEM, for all considered techniques. All units are expressed in meters.

As this work describes a method to adjust TRN to a large DEM, comparing the results given by the proposed pipeline with classic techniques becomes of interest.

Using the two models presented in Section 5.1, two experiments were designed in which four levels of detail for both DEMs are calculated, each one with a resolution 2 times lower than the immediate most detailed level, for our proposed pipeline and other reference methods. The generated meshes have been compared to the DEM and also between them, calculating their vertical accuracy.

These new experiments differ in how subsequent levels of detail are calculated via the same adjustment methods. For the first one, all levels of detail are calculated using a downsampled version of the DEM as initial approximation. The different methods used in this experiment are the linear interpolation (LI), LSQR (LS) and the proposed pipeline (PI). In the second one, the result of the generation of the precedent level of detail has been used instead as initial approximation. The methods for this case are labeled as Progressive LSQR (PLS) and Progressive Pipeline (PPI).

Regarding the LI method, it is important to clarify that the value of an intermediate point is computed as the bilinear interpolation of the four neighbouring data points. In contrast, the LSQR-based method obtains the optimal values for each vertex by iteratively approximating the solution of a linear system of equations. The LSQR implementation of Matlab has been set to seek a tolerance of 10^{-6} meters in the achieved solution, within a maximum of 300 iterations. As the experimentation shows, however, that number of iterations is never met whereas the total error is several orders of magnitude greater than the tolerance, indicating that the LSQR method finds itself stalled at undesired local minimums.

A first analysis of the results was done and presented in Figures 5 and 6, focusing on the resemblance between the meshes and the DEM. It is observed that the use of PI or PPI gives, for all levels, TRN with lower mean errors related to the DEM than the ones generated with the classic methods. Some examples are shown in Figures 11 and 12. They have also maximum errors similar to the ones obtained using the linear interpolation. On the other hand, LS and PLS only outrun the linear interpolation, in both maximum and mean error, for the TRNs of the coarsest levels of detail (3-4).

It is also observed that, for both scenarios in this experiment and all the tested techniques, the maximum error of a given TRN versus the DEM is always over 250 meters, with some cases presenting errors of 463 meters. By carefully analyzing those errors, it was discovered that all tested methods find difficulties in modeling large cliffs, as it is seen in Figure 10. The larger space between points in a regular grid implies that some sea points adjacent to the actual cliff will have a great height value in the TRN, thus, large errors will be produced. Moreover, line-shaped artifacts have been ob-

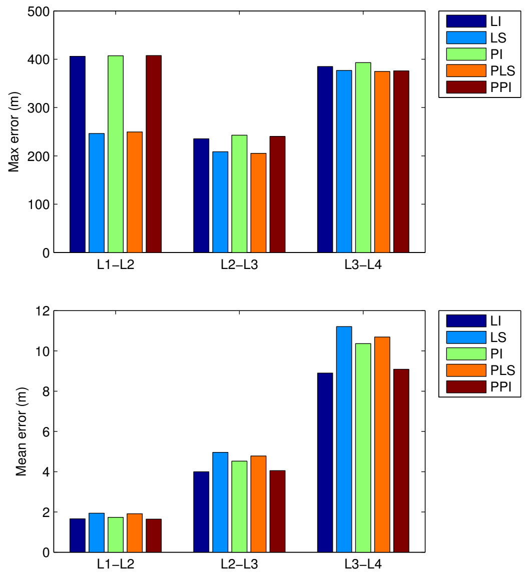
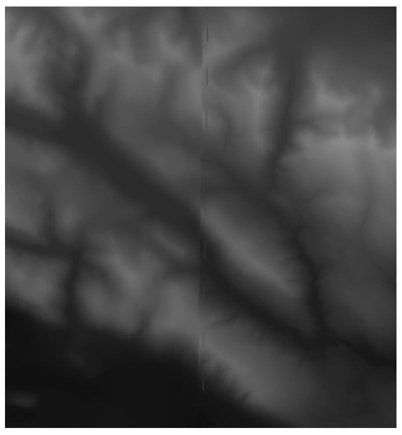
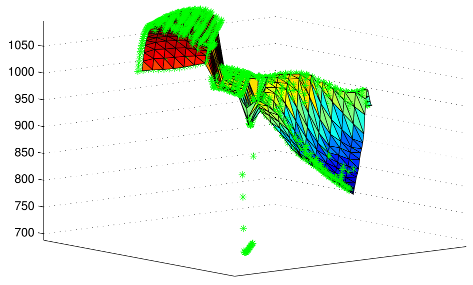


Figure 7: Maximum and mean errors between two consecutive levels of detail of the Puget Sound model, for all considered techniques. All units are expressed in meters.

served in some regions of the Puget Sound model, as it is presented in Figure 9. Variations on the height of the affected vertices can be of hundreds of meters compared with the surrounding regions, making hard the adjustment of TRNs from the model.



(a) Line-shaped artifacts in a DEM region.



(b) TRN compared with original DEM in an affected area.

Figure 9: An example of line-shaped artifacts in regions of the Puget Sound model.

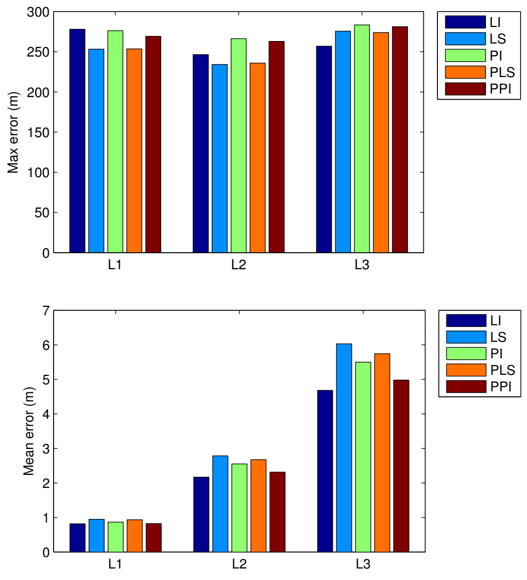


Figure 8: Maximum and mean errors between two consecutive levels of detail of the Tenerife model, for all considered techniques. All units are expressed in meters.

The second analysis compares the resulting levels with their inmediate precedent and its results are given in Figures 7 and 8. In general, it is seen that the linear interpolation provides the best results in both models. Computing TRNs directly from the DEM using LS, PLS or PI becomes inadvisable for visualization pur-

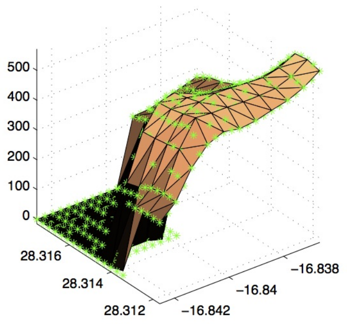


Figure 10: A cliff showing high errors in Tenerife DEM. Green markers represent DEM points to be confronted with TRN vertices.

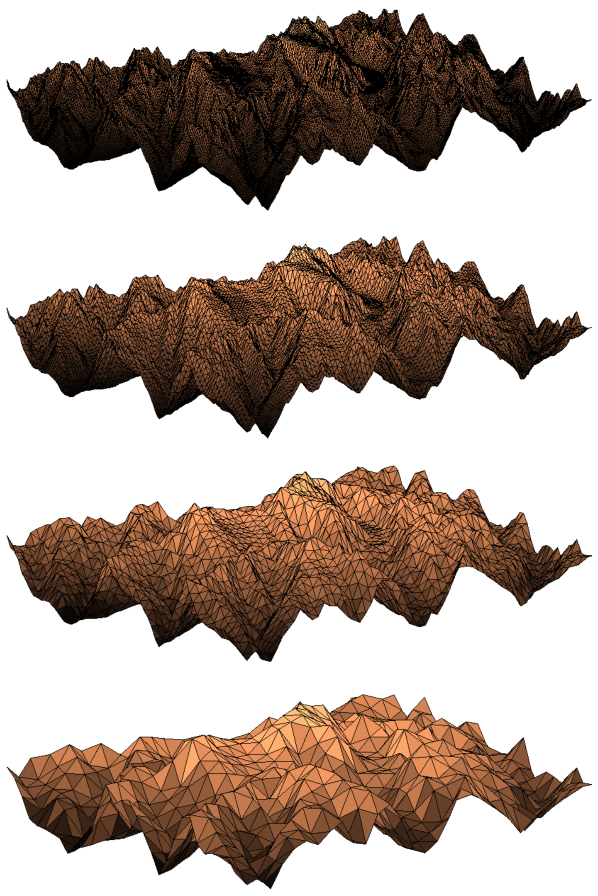


Figure 11: Area surrounding the highest point in the Puget Sound model, for all levels of detail generated using the proposed pipeline.

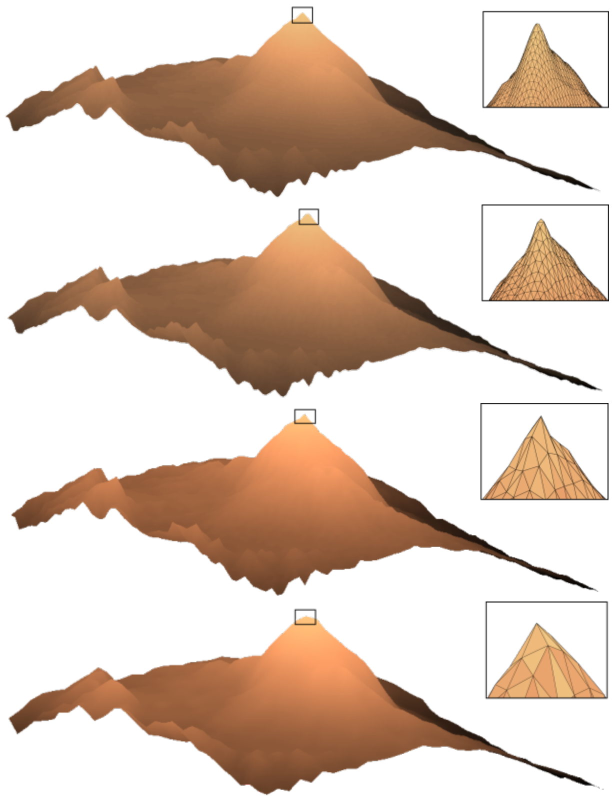


Figure 12: Results of the proposed pipeline for different levels of detail. In this case, the highest mountain of the Tenerife DEM (El Teide) is represented.

poses since it produces much higher errors between two consecutive levels of detail, unless a popping reduction technique such as geomorphing is applied to make smoother transitions. However, the use of PPI achieves better inter-level mean errors than the classic methods for detailed TRNs and only slightly worse mean errors for the coarsest levels. As this technique also allows to generate more accurate TRNs than the linear interpolation, as it was commented in the first analysis, the PPI method is also competitive for the adjustment of terrain models.

6 CONCLUSIONS

The present work has explored a way to adjust TRNs to better fit large DEMs. It consists in solving a linear system of equations that represent the height error of our representation. A technique to solve the linear system, such as LSQR, is used to approximate the system solution, and a later vertex refinement stage narrows down the mesh error.

The local refinement algorithm has an initial drawback due to large execution times when it is applied to large grids, but experimentation demonstrates that computational needs can be reduced by splitting the TRN in smaller windows without affecting the final result of the process.

The proposed pipeline was tested, generating meshes that fit the Puget Sound and Tenerife elevation models with promising results. Every mesh adjusted using our pipeline fits better the model than ones resulting of using classic methods as linear interpolation, without increasing heavily the difference between every level of detail. This fact suggests the pipeline can be used to generate meshes in order to visualize terrain in applications like virtual globes.

Future work will be focused on testing the pipeline versus other interpolation techniques in different scenarios and accelerating the local refinement stage. Moreover, a GPU implementation of the technique is also thought as future line of work.

7 ACKNOWLEDGEMENTS

The first author wants to thank Agencia Canaria de Investigación, Innovación y Sociedad de la Información, and the European Social Fund, for the grant “Formación del Personal Investigador-2012” that made possible this work.

The fourth author wants to thank Universidad de Las Palmas de Gran Canaria for its grant 'Programa de personal investigador predoctoral en formación 2015', which made possible this work.

8 REFERENCES

[Hop98a] Hoppe, H. Smooth View-dependent Level-of-detail Control and Its Application to Terrain Rendering, in Conf. proc VIS '98, Research Triangle Park, North Carolina, USA, IEEE Computer Society Press, pp. 35-42, 1998

[Els99a] El-Sana, S., and Varshney, A. Generalized View-Dependent Simplification. Computer Graphics Fourm, 1999.

[Pup96a] Puppo, E. Variable Resolution Terrain Surfaces, 1996.

[Lin96a] Lindstrom, P., and Koller, D., and Ribarsky, W. , and Hodges, L.F., and Faust, N., and Turner, G.A. Real-time, Continuous Level of Detail Rendering of Height Fields, in Conf. proc SIGGRAPH '96, ACM Press, pp. 109-118, 1996.

[Paj98a] Pajarola, R. Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation,in Conf. proc VIS '98, Research Triangle Park, North Carolina, USA, IEEE Computer Society Press, pp. 19-26, 1998

[Paj07a] Pajarola, R., and Gobbetti, E. Survey of Semi-regular Multiresolution Models for Interactive Terrain Rendering. Vis. Comput., vol. 23, pp. 583-605. Springer-Verlag New York, Inc. , July 2007.

[Yus11a] Yusov, E., and Shevtsov, M., High-Performance Terrain Rendering Using Hardware Tessellation., Journal of WSCG, vol. 19, pp. 85-92, 2011

[Kan15a] Kang, H., and Jang, H. , and Cho, C.S., and Han, J. Multi-resolution Terrain Rendering with GPU Tessellation. Vis. Comput., vol. 31, pp. 455-469, Springer-Verlag New York, Inc. , April 2015.

[Agu05a] Aguilar, F.J., and Aguera, F., and Aguilar, M.A., and Carvajal, F. Effects of terrain morphology, sampling density, and interpolation methods on grid DEM accuracy. Photogrammetric Engineering and Remote Sensing, vol. 71, pag 805-816, American Society for Photogrammetry and Remote Sensing, 2005.

[Cha06a] Chaplot, V., and Darboux, F., and Bourennane, H., and Leguédois, S., and Silvera, N.,and Phachomphon, K. Accuracy of interpolation techniques for the derivation of digital elevation models in relation to landform types and data density. Geomorphology, vol. 77, pp. 126-141, 2006.

[Qua16a] Quaglino, A., and Krause, R. Towards a multigrid method for the minimum-cost flow problem. arXiv preprint arXiv:1612.00201, 2016.

[Pai82a] Paige, C.C., and Saunders, M.A. LSQR: An algorithm for sparse linear equations and sparse least squares. ACM transaction on mathematical software, vol. 8, pp. 43-71, 1982.

[Peu78a] Peucker, T.K., et al. The triangulated irregular network. In Amer. Soc. Photogrammetry Proc. Digital Terrain Models Symposium. 1978. p. 532.

[Lue02a] Luebke, D., and Watson, B., and Cohen, J.D., and Reddy, M., and Varshney, A. Level of Detail for 3D Graphics. Elsevier Science Inc., 2002.

[Agu03a] Aguero, J. C., and Feuer, A., and Goodwin, G. C. Terrain Modelling via Triangular Regular Networks. MODSIM 2003, vol. 33, 2003.

[Pug00a] Finlayson, D., and Haugerud, R., and Greenberg, H., and Logsdon, M. : Combined Bathymetry and Topography DEM of Western Washington State. School of Oceanography of the University of Washington. October 2000. Web resource: http://www.ocean.washington.edu/data/pugetsound (Last accessed: 2017.02.17)

[Los04a] Losasso, F., and Hoppe, H. Geometry Clipmaps: Terrain rendering using Nested Regular Grids. ACM Transactions on Graphics, vol. 23, n. 3, pp. 769-776. August 2004.

[Liv09a] Livny, Y., and Kogan, Z., and El-Sana, J. Seamless patches for GPU-based terrain rendering. The Visual Computer, vol. 25, no 3, p. 197-208. 2009

[Tru12a] Trujillo, A., Suárez, J.P., De La Calle, M., Gómez-Deck, D., Santana, J.M. An open source virtual globe framework for iOS, Android and WebGL compliant browser. In Conf.Proc. COM.Geo'12, New York, NY, USA; ACM Press; 212, p. 22:1 - 22:10

[Sua03a] Suárez, J.P., Plaza, A. Refinement and hierarchical coarsening schemes for triangulated surfaces. Journal of WSCG., vol. 11, no. 1-3, 2003.

[Sua15a] Suárez, J.P., Trujillo, A., Santana, J.M.,De La Calle, M., Gómez-Deck, D. An efficient terrain Level of Detail implementation for mobile devices and performance study. Computers, Environment and Urban Systems, vol. 52, pp. 21-33. 2015.