Efficient visualization of horizons with faults in seismic data

Fernando V. da Senhora* (TecGraf), Jéferson R. P. Coêlho (TecGraf), Marcelo Gattass (TecGraf)

Abstract

The interactive visualization of surfaces representing the horizons in massive seismic data is a problem due to the associated computational cost. Often these surfaces are represented by large triangular meshes with a vertex in each sample. This representation consumes significant part of the memory and the display processing time, restricting the size of data that can be viewed. However, horizons can be generally represented by surfaces that contain low varying regions that can be represented by a few triangles without incurring in significant errors. This work presents an efficient method for generating adaptive triangular meshes for representing seismic horizons with or without faults with waste. The method produces a compact triangular mesh that is only refined where the variation of the surface requires more triangles. The proposed method supports not only the generation of a mesh to represent an already interpreted horizon by points, but it can also be used to interactively adjust the horizon as the interpreter adds new points. Results are shown to corroborate the method.

Introduction

The visualization of horizons on conventional hardware presents certain challenges due to the large number of data they contain. The direct approach of displaying every single point collected is not always possible or desirable, since it would require lots of memory that could be used for other tasks in the processing of seismic data. Besides that, real data contain considerable noise in most applications. Therefore, it is desirable to create a smooth surface with the least possible number of elements. A mesh that approximates this horizon can represent the same information with less data, making it easier to visualize it, perform simulations, or any other type of analysis that may be necessary.

The main benefit of this method is its ability to represent massive horizons spending little memory quickly and efficiently. The mesh is represented by a data structure called Corner Table (Coêlho and Gattass 2013). This requires little storage space, but still provides fast operations on its elements, in addition to supporting adaptive subdivision.

The method only covers the cases where the data points are a scalar function of the position in a plane. This type of data can represent quite well the topography of a region, height maps or simple objects. The purpose of this work is to deal with horizons. Cases of salt domes and other more complex formations will not be addressed.

It is common that these geological layers present faults that appear as discontinuities on the surface. To represent this phenomenon we need a mesh that is both smooth and discontinuous at the same time. This paper assumes that the faults are already known and does not bother to locate them. Once the faults are known, it has as an input parameter a mesh, in which the discontinuities are inserted in its topology. Having this information the mesh is refined where it is necessary to achieve a final smooth and precise result. The input mesh can be coarse provided that it meets the discontinuity.

In order to evaluate the performance of the proposed method, we used the data from the volume of the Netherlands offshore F3 block downloaded from the Opendtect website.

In the next section we present some research done on the subject. The details of the method are described in Proposed Method section. We present the results in the Results section and finally the Conclusion section summarizes the work ideas.

Related Work

Most research done in the field is dedicated to treat more general surfaces than those addressed in this work. Many studies have been made in the modeling of complex objects (such as salt domes) which cannot be represented by a function, and therefore are not covered by the proposed method. The best results for this kind of problem are obtained using implicit functions and the method developed in Mallet (2002) called Discrete Smooth Interpolation. Although effective the method is rather complex and its computational cost is high.

The purpose of this work is to deal with horizons. In such cases the use of the methods described above is unnecessarily complicated and expensive.

Limited to these cases, there are several approaches described in the literature, each with its merit.

A group of works uses its own points to represent the surface. Gregorski et al. (2000) for instance builds B-splines to describe the points and thus generates a smooth surface without producing a mesh of polygons. Although interesting these methods require a dense set of points and present no noise control.

Another approach is the explicit methods. They use the data points as vertexes of the resulting mesh. These are generally based on Delaunay Triangulation and Voronoi diagrams. Among them stands out the α-shape method Edelsbrunner and Mücke (1994). Again the main problem of these methods is the smoothness and noise. In Mederos et al. (2005), the crust algorithm was modified to deal with noise, but the result is not smooth.

Our algorithm falls into the category of implicit methods. This works by defining a function f (x, y, z) whose value is equal to the distance of the point (x, y, z) to the surface, and zero if the point (x, y, z) is on the surface. The problem is then to find iso-curve f (x, y, z) = 0 of the function.

The most common way to build this function is using radial functions and B-Spline, both guarantee smoothness. Most of these however requires the data to be regularly displayed and have difficulty dealing with discontinuities (Gregorski et al., 2000). Arge and Floater (1994) present a solution using radial functions dealing with discontinuities and sparse data in square meshes but regions with no information should be handled separately and their method does not deal with noise in the data.

In the general 3D case approached by Frank et al. (2007) the problem results in a nonlinear system, and DSI is used to obtain a solution.

Because we restrict our problem to points representing a function and have the distance from one point to the surface as simply being the vertical distance, the implicit function is much simpler to be built. It is worth noting that, because of this, we only modified the height of each vertex and not its position in the xy plane. With these simplifications we have a linear system. The authors of Hjelle and Dæhlen (2006) describe this method in the book Triangulation and Applications

Least Square

The least squares method used is well known and established in literature. It is based on the minimization of the squared error to find the best fit for a set of data.

The problem becomes simply finding the heights of vertex that minimize the error of the triangulation in respect to the data points (x_k, y_k, z_k). Since we assume the coordinates (x, y) of each vertex are fixed, we define the error as the vertical distance from the point to the mesh.

Proposed Method

The studied method is summarized in six major ideas that guarantee a fast and efficient solution, even with a large dataset. They are:

1. Implicit Function: The surface is constructed as a combination of linear functions.
2. Least Squares: We use the least square method to obtain a system of linear equations to find the parameter that minimize the error of the implicit function.
3. Smoothing Terms: We add a term that ensures a unique and smooth solution.
4. System Solver: The linear system of equations is solved by an iterative method.
5. Local Refinement: The mesh is refined only where the error is greater than a specified threshold.
6. Progressive increase in the number of points: Optional step aiming efficiency.

These six processes are performed repeatedly until the desired condition is achieved, as described in the Final Algorithm section.

In the next sections, these ideas are explained in the order in which they appear. After that, an overview of the algorithm and the treatment of discontinuities is discussed.

Implicit Function

The first step is the discretization of the surface in a number n of nodes. These nodal points are the vertexes of the triangulation. Each vertex will have a basis function associated to it. The basis functions are N_i , and v_i , which are the associated vertex. These functions have the following properties:

- They are linear in their domain.
- They have value 1 in the position of its associated vertex.
- They are zero in the adjacent vertexes and in the opposite edges.
- They are zero outside the triangles that contain them.

Figure 1 shows a graphical representation of a basis function.

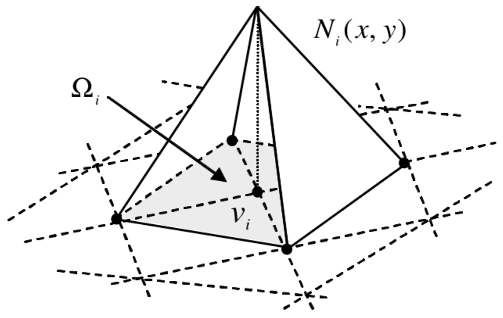

Figure 1: Figure taken from Hjelle and Dæhlen (2006) representing the basis function used.

The linear combination of these basis function is a representation of the surface triangulation. It can be written as:

$$f(x,y) = \sum_{i=1}^{n} c_i N_i(x,y) \tag{1}$$

The variables $c_1,..,c_n$ are coefficients of f , and represent the value that the mesh assumes in the position of the vertex. This sum has at most three terms different than zero for each point (x,y) because of the basis functions properties. This implies that the surface is defined locally. Changes in a region of the surface, only affect a limited neighborhood.

Least Squares Application

By applying the least square method, we obtain a linear system whose solution is the best approximation of the surface to the dataset.

The locality properties of the implicit function guarantee that our system of equations will be sparse and symmetric. The matrix is also always positive semi-definite.

Smoothing Term

Often, the triangulation generated is not smooth, especially when the data is subjected to noise. Crests and troughs may appear due to problems of "over-fitting", which is undesirable.

Moreover, there are situations in which a region of triangles has no internal points. To be able to ensure that the system has only one solution, it is necessary to have at least one data point within at least one of each triangle vertex. The region Ω, in Figure 2, for example, does not satisfy this condition. Vertexes inside that region have no restrictions imposed to them by the data, and their heights can have any value without affecting the error of the triangulation. Thus, the system is under-defined and cannot be solved uniquely.

To mitigate this issue, we added a penalty term to the system. This term is directly related to the triangulation. What it does is entangle all the triangles of the mesh, thereby creating restrictions usually related to some energy functional.

Two penalty functionals were implemented. They are based on the first and second derivative, respectively.

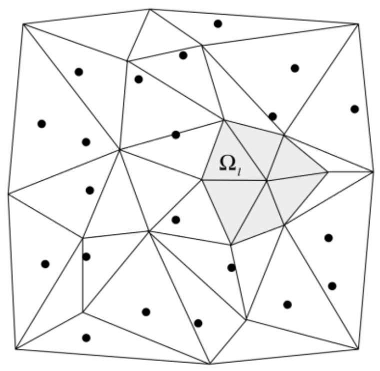

Figure 2: Figure taken from Hjelle and Dæhlen (2006). Triangulation scheme on data points.

They are approximated in the discrete case. Both satisfy the above conditions and also ensure a linear system with a single solution. They are:

1. Membrane Energy: Based on the first derivative, it interprets the surface as a membrane.
2. Thin-Plate Energy: Based on the curvature (second derivative), it interprets the surface as a thin sheet.

The best results were obtained using a combination of the two. But it is interesting to set a different weight to each one and so be able to adapt the surface as needed. That is, if the surface prioritizes the variation of height or curvature.

A more detailed description of the least squares linear system and penalty methods can be found in Hjelle and Dæhlen (2006).

System Solver

Once we have the system ready, it is just a matter of finding a good solution technique for this kind of problem. We chose the Conjugate Gradient method. Being an iterative method, it can be applied to sparse systems that are too large to be handled by direct methods, such as Cholesky decomposition. Its performance can be enhanced using a good initial solution.

Local Refinement

A multilevel triangulation is used so we can achieve a final mesh with the smallest possible number of triangles without losing significant information. The initial mesh is coarse and it is refined locally where the error is larger than a specified threshold.

A triangle is subdivided if there are any data points in its interior with error greater than a defined limit.

This is done recursively until a desired condition is achieved. In each iteration, we must solve a least squares system. However, using the previous solution as an initial guess in the conjugate gradient method only a few iterations are necessary. Often one iteration is enough. The subdivision scheme used was the one described in Coêlho and Gattass (2013).

Progressive increase in the number of points

Starting with a coarse mesh, working with a lot of points is not beneficial. A similar result is obtained when approaching two triangles to a million points or to ten thousand. Therefore, we start with a small group of data points of the set and increment this number until we are using all the data. This ensures superior speed while solving the problem.

The question is whether the final mesh will be equivalent. Tests done with real data indicate that, with this progressive increase, the average error is slightly higher in some situations. See more on this in the Results Section. However, the value was small even in the worst cases and well below the specified error limit. For most applications, therefore, this difference is irrelevant.

Final Algorithm

Once these ideas are clear the algorithm is shown in Figure 3.

```
Data: Initial Mesh and Dataset
do
    Increase the number of data points;
    while Exist any triangle with error greater than the treshold do
        Least Square system;
        System solver;
        Error evaluation;
        Local refinement;
    end
while Number of data points used $\neq$ Total number of data points;
```

Figure 3: Final Algorithm

Faults, gaps and discontinuities

Surfaces can be complex and have discontinuous regions or regions where it is not desirable to have representation. Horizons, for example, often have several faults in its domain. The result must comply with these conditions.

It is possible to reproduce these faults using a pre-processed mesh with discontinuity represented in its topology as input. The adaptation of the mesh to discontinuities is a well-studied problem and is beyond the scope of this article. The method is limited to approximating the surface once the faults are interpreted. In this work, it is assumed that the faults are already known.

Because of the progressive refinement the initial mesh can be as coarse as it is necessary provided that it meets the limitations wanted in the final result. No modification of the algorithm is needed. The reason for this is the form of the smoothing parameter and the implicit function. Both operate on the topology and the refinement propagates this structure. Thus the discontinuities are satisfied naturally by the method.

Results

In our tests, we used the data from the volume of the Netherlands offshore F3 block downloaded from the Opendtect website. These data have noise, regions without information and discontinuities. The method proved itself to be robust and efficient, even when dealing with a large number of points. Tables 1 and 2 have the number of generated triangles, the maximum error, the average error, the solution time and the ratio of the memory used by the method proposed against the traditional method. Note that because of the noise, it is not desirable to have zero error. The restriction of smoothness prevents all points to be interpolated. The limit error imposed was 4.00, and it represents the discretization of the time dimension of the data used.

Each file has close to 1050625 data points. Variables of type float were used to store the geometry and variable of type in the topology. In traditional representation, data is arranged in a regular grid so it can be represented by a nxm matrix. Therefore, each point only need one coordinated to be described, as the spacing between samples is constant. The traditional triangulation uses 3 variables for each triangle. So each tested surface occupies close to 29 Megabytes of memory. In our method, the vertexes need 3 coordinates to be described and triangulation requires 6 variables, because of the Corner Table structure. Even with these conditions, it was possible to reduce memory necessary, since the number of elements used is much lower.

Table 1 shows results using progressive increase of the number of points, while Table 2 uses the total number of points from the beginning. Figures 4 and 5 are the reconstructed data surface "Map 1" described in the Tables, Figures 6 and 7 are the surface "Map 5" and Figure 8 shows the approximation of a region with no data.

Table 1: Performance Data of the algorithm with gradual incrementation of points

Gradual incrementation of the number of points					
Surface	Triangles	Max Error	Average Error	Time (s)	Memory Ratio
Map 0	151815	7.88	0.716	24	15.52%
Map 1	240388	10.74	0.821	34	24.57%
Map 2	53482	11.08	0.630	14	5.47%
Map 3	76811	7.68	0.681	12	7.86%
Map 4	179621	9.65	0.710	23	18.37%
Map 5	50265	17.5	0.671	11	5.14%
Map 6	449584	5.65	0.840	34	45.96%
Map 7	151733	5.21	0.830	18	15.52%

Table 2: Performance Data of the algorithm without gradual incrementation of points

Using All Points					
Surface	Triangles	Max Error	Average Error	Time (s)	Memory Ratio
Map 0	149270	10.82	0.678	28	15.26%
Map 1	240095	10.17	0.756	37	24.54%
Map 2	53569	10.82	0.615	14	5.48%
Map 3	77255	18.04	0.679	17	7.90%
Map 4	177918	12.56	0.714	25	18.20%
Map 5	49763	16.31	0.661	16	5.09%
Map 6	448014	13.32	0.841	44	45.80%
Map 7	155839	12.17	0.816	26	15.94%

Comparing the two percentually in Table 3, it is noticeable the reduction of the execution time in most cases. In the worst case the time remained the same. Although there was a small increase in the average error, this was far below the established limit. However, the maximum error tends to decrease as well as the number of bad triangles, with the gradual increase of points, which is desirable. This is because of the imposed stopping conditions. Negative values indicate an improvement of the incremental method, and positive values indicate worsening.

Table 3: Algorithm performance comparison with and without gradual increase of data points.

Comparison					
Surface	Triangles	Max Error	Average Error	Time (s)	Memory Ratio
Map 0	1.70%	-27.2%	5.61%	-14%	1.70%
Map 1	0.12%	5.6%	8.49%	-8%	0.12%
Map 2	-0.16%	2.4%	2.36%	0%	-0.16%
Map 3	-0.57%	-57.4%	0.36%	-29%	-0.58%
Map 4	0.96%	-23.2%	-0.45%	-8%	0.96%
Map 5	1.01%	7.3%	1.56%	-31%	1.01%
Map 6	0.35%	-57.6%	-0.11%	-22%	0.35%
Map 7	-2.63%	-57.2%	1.51%	-30%	-2.64%

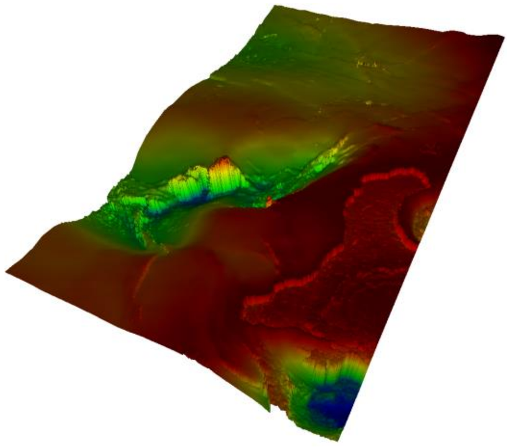
Figure 4: Surface "Map 1" generated by the proposed method.

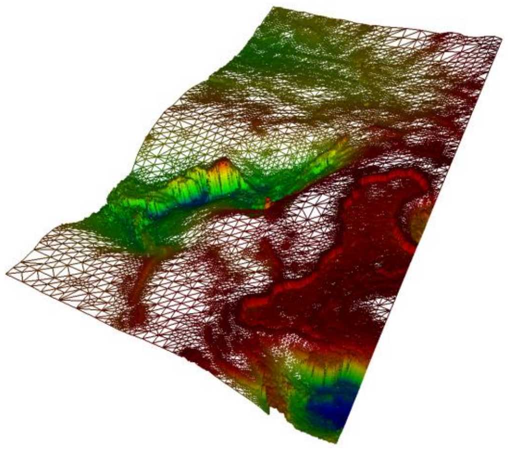
Figure 5: Mesh associated with "Map 1" generated by the proposed method.

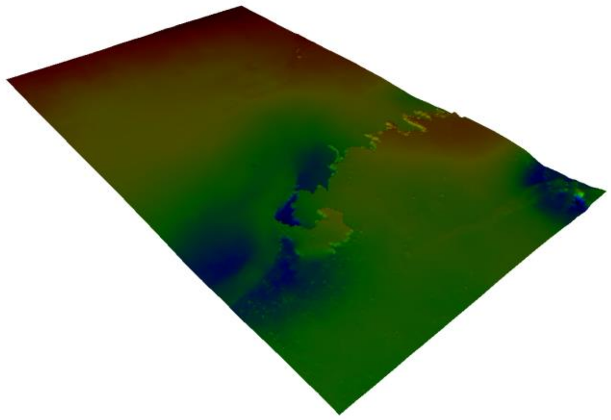
Figure 6: Surface "Map 5" generated by the proposed method.

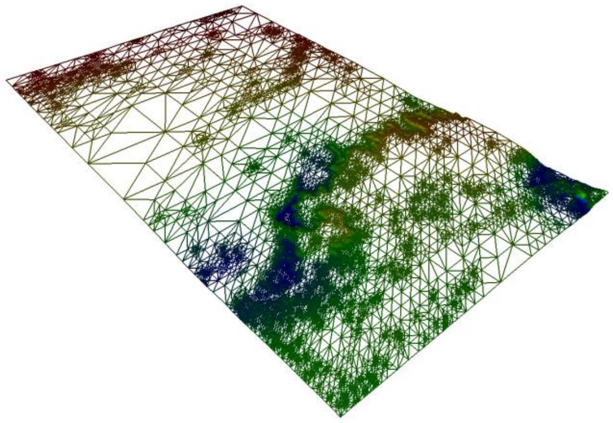
Figure 7: Mesh associated with "Map 5" generated by the proposed method.

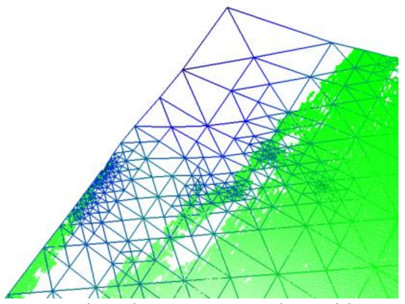
Figure 8: Mesh approximation over a region with no data.

As for gaps and faults, no change in the algorithm itself is necessary. Having a mesh with a topology structure already adapted, satisfactory results are shown in Figures 9 and 10.

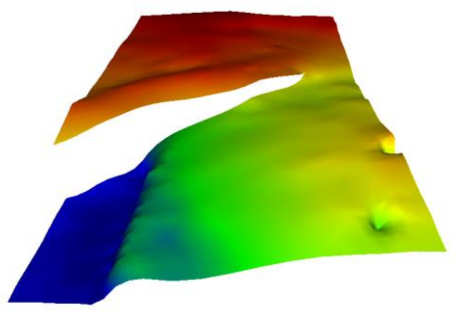

Figure 9: Surface generated over fault.

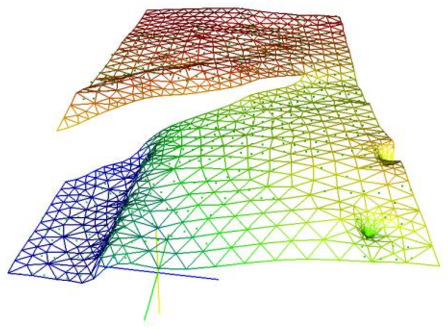

Figure 10: Mesh generated over fault.

Conclusion

The algorithm is simple, efficient, computationally light and can be performed iteratively with a considerable number of points. It is based on the construction of an implicit function and uses the least squares method to generate a linear system. The smoothing term ensures that the result is smooth, even when noise is present and it extrapolates the solution to regions without data. Discontinuities are not a problem if they are treated in pre-processing of the initial mesh, and the local refinement guarantees a triangulation with a small number of elements.

Good results were obtained using real data under adverse conditions. The method proves to be useful to model seismic horizons, especially those with many irregularities. The construction of the surface can be made as the data points are interpreted, also providing the user with a more accurate intuition in the initial stage of the interpretation. Furthermore, it was possible to represent the same surface with a much smaller number of elements, thus saving computational resources as proposed in this work.

References

Arge, E. and Floater, M. (1994). Approximating scattered data with discontinuities, numerical algorithms 8.

Coêlho, J. and Gattass, M. (2013). Uma solução eficiente para subdivisão de malhas triangulares. Master’s thesis, PUC-Rio.

Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes.

Frank, T., Tertois, A.-L., and Mallet, J.-L. (2007). 3d-reconstruction of complex geological interfaces from irregularly distributed and noisy point data. Computers and Geosciences, 33(7):932–943.

Gregorski, B. F., Hamann, B., and Joy, K. I. (2000). Reconstruction of b-spline surfaces from scattered data points. In In Proc. Computer Graphics International 2000, pages 163–170.

Hjelle, O. y. and Dæhlen, M. (2006). Triangulations and Applications (Mathematics and Visualization). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Mallet, J.-L. (2002). Geomodeling (Applied Geostatistics Series). Oxford University Press, USA, 1 edition.

Mederos, B., Amenta, N., Velho, L., and de Figueiredo, L. H. (2005). Surface reconstruction from noisy point clouds. In SGP ’05: Proceedings of the third Eurographics symposium on Geometry processing, pages 53+, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.