Applied Mathematics and Computation 216 (2010) 2937–2946

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A model to simplify 2D triangle meshes with irregular shapes

L. Tortosa *, J.F. Vicent, A. Zamora

Departamento de Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante, Ap. Correos 99, E–03080 Alicante, Spain

ARTICLE INFO

Keywords:
Algorithm analysis
Mesh simplifications
Artificial neural networks
Triangulation
GNG3D model
Self-organizing systems

ABSTRACT

We present a 2D triangle mesh simplification model which is able to produce high quality approximations of any original planar mesh, regardless of the shape of the original mesh. This method consists of two phases: a self-organizing algorithm and a triangulation algorithm. The self-organizing algorithm is an unsupervised incremental clustering algorithm which provides us a set of nodes representing the best approximation of the original mesh. The triangulation algorithm reconstructs the simplified mesh from the planar points obtained by the self-organizing training process. Some examples are detailed with the purpose of demonstrating the ability of the model to perform the task of simplifying an original mesh with irregular shape.

1. Introduction

Research on mesh simplification in Computer Graphics and Scientific Visualization has led to a substantial number of methods within the last two decades. An exhaustive description of this field is beyond the scope of this paper; one can refer to the many surveys available (see, for e.g., [6]). However, it must be emphasized that mesh generation in regions in Euclidean space is a central task in computational science, and especially for commonly used numerical methods for the solution of partial differential equations, e.g., finite element and finite volume methods.

In addition, several geometric problems for sets of n points require to triangulate planar points sets [9]. This encourages motivates an increased attention in problems related to the triangulation of planar point sets and 2D triangle meshes, as we can see in recent works [16,4,3,5,14].

The most popular triangulation method that uses the xy projections of the input points is Delaunay triangulation [7]; it is a purely two-dimensional method. Delaunay triangulation finds the triangulation that maximizes the minimum angle of all triangles, among all triangulations of a given point set.

The typical surface models handled by contemporary computer graphics applications have millions of triangles. Mesh simplification has emerged as a critical step for handling such huge meshes. The problem of approximating a given input mesh with a less complex but geometrically faithful representation is well-established in computer graphics. Level-of-detail representations figure prominently in real-time applications such as virtual reality, terrain modeling and scientific visualization.

Over the last decades a tremendous amount of work has been done on mesh simplification. Most of the techniques or algorithms proposed to accomplish this objective are based on reducing the mesh complexity either by merging/collapsing elements or by re-sampling vertices. A good overview of the state of the art in this field is [12]. Simplification strategies may be broadly grouped into two categories: local strategies that iteratively simplify the mesh and global strategies that are applied to the input mesh as a whole.

* Corresponding author.
E-mail addresses: tortosa@dccia.ua.es, tortosa@ua.es (L. Tortosa), jvicent@ua.es (J.F. Vicent), zamora@ua.es (A. Zamora).

0096-3003/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2010.04.005

A recent topic in mesh simplification is the preservation of topological characteristics of the mesh and of data attached to the mesh [17].

Self-organizing networks are able to generate interesting low-dimensional representations of high-dimensional input data. The most well-known of these models is the Kohonen Feature Map [11]. It has been used in the last decades to study a great variety of problems such as vector quantization, biological modeling, combinatorial optimization and so on.

We present in this paper a new topology preserving simplification model for 2D triangle meshes. This model can be described in two phases: a first phase given by a self-organizing algorithm that provides us a cloud of points representing a distribution of the nodes in the simplified 2D mesh, and a second phase given by a triangulation algorithm, whose primary purpose is to reconstruct the 2D simplified mesh from the cloud of points obtained in the self-organizing algorithm. Throughout the article we will refer indistinctly to nodes or vertices.

2. The simplification model

A self-organizing network consists of a set of neurons arranged in some topological structure which induces neighborhood relations among the neurons. An n-dimensional vector is attached to every neuron, which determines the specific n-dimensional input signal to which the neuron is maximally sensitive. By assigning to every input signal the neuron with the nearest reference vector, a mapping is defined from the space of all possible input signals onto the neural structure.

Self-organizing networks learn in an unsupervised manner from a stream of input signals. For each input signal, the neuron with the nearest reference vector is determined, called the *winner unit*. Then, the reference vectors of the winner unit and some of its topological neighbors are moved towards the input signal. The adaptation of topological neighbors distinguishes self-organization (*winner takes most*) from competitive learning, where only the winner is adapted (*winner takes all*).

These particular unsupervised networks have been applied to the problem of surface reconstruction. Yu [18] introduced a novel technique for surface reconstruction from unorganized points by applying Kohonen's self-organizing map. The topology of the surface is predetermined, and a neural network learning algorithm is carried out to obtain correct 3D coordinates at each vertex of the surface.

The model we present in this section is able to simplify any 2D triangle mesh, with the primary characteristics summarized in the following:

- It allows to obtain a planar point set that will constitute the vertices of the simplified mesh. These points are obtained using a neural network algorithm; more exactly, an incremental clustering algorithm as it will be described in the following.
- It performs a triangulation process based on a comparison between the original triangle mesh and the reduced set of vertices. After this process, a new simplified two-dimensional mesh is generated with a similar shape as the original one.
- It allows to approximate or simplify any 2D triangle mesh, even presenting irregular shapes.

The model may be subdivided into two different phases. The first one consists of getting the set of vertices to simplify the original mesh while the second phase consists on the triangulation of the vertices obtained in the previous step with the aim of generating the final simplified mesh.

In the first phase, the objective is to find a set of vertices with the basic property that the geometric distribution of the new vertices minimizes the distance between them and the original planar points in the original triangle mesh. For this purpose, an algorithm based on the concept of self-organizing maps introduced by Kohonen [11] is used.

The central idea of the algorithm coincides with the model GNG3D used to reduce 3D objects (see [1,15]); however, some deep modifications and simplifications have been introduced in the optimization algorithm of the GNG3D model, in order to adapt it to the case of working with point sets in the plane. The number of parameters that control the training process has also been reduced.

We consider, as a starting point, a 2D triangle mesh consisting of.

- a set $A = \{n_1, n_2, \ldots, n_N\}$ of vertices or nodes,
- a set $T = \{t_1, t_2, \ldots, t_L\}$ of triangles among node pairs.

We can say that the set T constitutes the set of triangles or faces that make up the original 2D mesh. Consequently, each element of the set T is given by

$$t_i = \{n_j, n_k, n_l\}, \quad n_j, n_k, n_l \in A.$$

It is important to point out that we are going to fix the number of vertices of the simplified mesh; therefore, if we set the number of vertices of the simplified mesh as M, then the objective is to find a set of vertices

$$K = \{k_1, k_2, \ldots, k_M\},$$

where k_i, for $i = 1, 2, \ldots, M$ are the new nodes in the simplified triangle mesh. Note that always $M < N$ and M is fixed. The set K may be computed by means of the following algorithm.

Phase 1: *the self-organizing algorithm.*

INIT: Start with M points $k_1, k_2, \ldots, k_M$ at random positions $w_{k_1}, w_{k_2}, \ldots, w_{k_M}$ in R^2. Initialize a local counter l_c to zero for every point or node k_i, for $i = 1,2,\ldots,M$. We set the maximum number of iterations as I_t.

1. Generate an input signal ξ that will be a random point $n_i \in A$ from the original mesh.
2. Find the nearest node s_1.
3. Find the second and third nearest nodes, s_2 and s_3 to the input signal.
4. Increment the local counter of the winner node s_1.
5. Move s_1 towards ξ by fractions ϵ_{win} respect to the total distance

$$\Delta w_{s_1} = \epsilon_{win}(\xi - w_{s_1}).$$

6. Move s_2 and s_3 towards ξ by fractions ϵ_n respect to the total distance:

$$\Delta w_{s_2} = \epsilon_n(\xi - w_{s_2}),$$
$$\Delta w_{s_3} = \epsilon_n(\xi - w_{s_3}).$$

7. Repeat steps 1 to 6 λ times, with λ an integer of order $O(10^2 \cdot N)$. We have the local counter vector $L_c = (l_c(k_1), l_c(k_2), \ldots, l_c(k_M))$. For $i = 1,2,\ldots,M$,
   - If $l_c(k_i) = 0$ then
     - We choose randomly a node n_j from the original mesh.
     - Make $k_i = n_j$.
   - If $l_c(k_i) \neq 0$ then continue.
8. Stop when the maximum number of iterations I_t has been reached.

This first part of the algorithm can be seen as a training process, where a set of nodes representing the new vertices of a planar mesh are obtained. So far, nothing about the triangles or faces of the original planar mesh has been mentioned. The central idea of Phase 1 may be considered as a generalization of the MacQueen's algorithm, which constitutes a process for partitioning an N-dimensional population into k sets on the basis of a sample, (see [8,10,13] for a more detailed description). The second part of the process of generation of the simplified mesh is ready to begin. Now, a triangulation process is developed with the aim of reconstructing the new mesh. Our proposal constitutes a post-process which uses the information provided by the self-organizing algorithm and the information about the nodes and triangles of the original mesh. This triangulation procedure can be summarized in the following algorithm.

Phase 2: *the triangulating process.*

INIT: Consider the set A of the original nodes, T the triangles of the original 2D triangle mesh, and K the set of the nodes obtained by the above self-organizing algorithm.

1. Associate each node of the original mesh with a node of the set K.For every n_i, for $i = 1,2,\ldots,N$, find $j \in \{1,2, \ldots,M\}$ such that

$$|w_{n_i} - w_{k_j}| \leqslant |w_{n_i} - w_{k_l}|, \quad l \in 1,2,\ldots,M,$$

where w_{n_i} represents the position of the node n_i. Save (n_i, k_j). We say that k_j is the node associated to n_i.
2. Change the nodes of the original triangles by their associated nodes. For every $t_i = \{n_{i_1}, n_{i_2}, n_{i_3}\} \in T$, substitute

$$\{n_{i_1}, n_{i_2}, n_{i_3}\} \to \{k_{j_1}, k_{j_2}, k_{j_3}\},$$

where $k_{j_1}, k_{j_2}, k_{j_3}$ are the associated nodes of $n_{i_1}, n_{i_2}, n_{i_3}$, with $j_1, j_2, j_3 \in \{1,2,\ldots,M\}$.
   - If $k_{j_1} \neq k_{j_2} \neq k_{j_3}$, then save $t_i = \{n_{i_1}, n_{i_2}, n_{i_3}\}$.
   - If $k_{j_1} = k_{j_2}$, or $k_{j_1} = k_{j_3}$, or $k_{j_2} = k_{j_3}$, then continue.
3. Graph the set

$$\mathscr{C} = \{t_i = \{n_{i_1}, n_{i_2}, n_{i_3}\}, \quad k_{j_1} \neq k_{j_2} \neq k_{j_3}\}.$$

4. If some node is isolated we add a new triangle, linking this node with their adjacent nodes.

We can summarize the process of triangulation, saying that we perform a comparison between the nodes of the original 2D triangle mesh with the nodes generated by the first algorithm, in order to determine the triangles that must be graphed in the new mesh, following a *learning shape* process.

3. Some aspects of the algorithms

Regarding the self-organizing algorithm described in the first part of the model, it must be pointed out that it is based on the GNG3D algorithm for 3D objects; however, although the central idea of the self-organizing models is shared, some basic features of the new algorithm should be discussed because it makes it quite different from the GNG3D algorithm.

Therefore, the basic features of the new self-organized algorithm may be summarized in the following items:

- Initially we take M nodes randomly as a starting point, which represents a quite different initial approach to that proposed by the GNG3D model, where the number of nodes of the final mesh was totally controlled by the user throughout the training process of the construction of the mesh. This is a critical aspect because it eliminates the entire process of adding and removing nodes. Therefore, we focus on the process of self learning the shape of the original mesh.
- Throughout this process we do not take into account the triangles or faces that form the initial mesh.
- The parameters involved in the training process are $\epsilon_{win}, \epsilon_n$ and λ. The parameter ϵ_{win} is related to the displacement of the winner node, while ϵ_n is related to the displacement of the neighbor nodes in the plane. The parameter λ is introduced in order to be sure that any node of the initial set of random points will stay isolated during the entire execution of the algorithm.

Some clarification is necessary regarding the parameter λ and the point 7 of the self-organizing algorithm. As we randomly chose the geometric positions of the initial M nodes (point 1 of the algorithm), it is possible that some of these initial points were generated in positions far away from the area covered by the original mesh. Accordingly, no signal will be near these points and, therefore, these nodes will never be *winners* in the training process. Consequently, its local counter will remain zero as the number of iterations increase. So, this is the way we have to detect when a node is outside the area of the initial triangle mesh. This is the reason for introducing the condition in the point 7 of the self-organizing algorithm. In this point, we replace the isolated nodes by other nodes but in the original mesh and the way to choose them is at random. This feature will be studied in more detail later when we present some examples of triangle mesh simplifications. Then, we will visualize how isolated vertices can arise when we try to simplify networks with very irregular shapes and the initial vertices are chosen randomly.

In the self-organizing algorithm the positions of the initial vertices are modified with the purpose of learning the shape of the original mesh. After this process we have no information about the triangles that make up the simplified mesh. We only have the geometric position of the new vertices; therefore, an efficient algorithm may be implemented to reconstruct the 2D mesh preserving the original shape.

The most popular triangulation method is the Delaunay triangulation, which basically finds the triangulation that maximizes the minimum angle of all triangles, among all triangulations of a given point set. We follow, as it has been described in the algorithm, a rather different approach. The idea underlying our approach is to establish a special equivalence relation in the original set of vertices. The equivalence relation consists of assigning to every node of the set A a node of the set K which is nearest to it. After this association, we will have some equivalence sets because some nodes of the original set A will have as a representative the same node of the set K (remember that the cardinal of K is much less that the cardinal of A). After this concordance process, we only have to determine the triangles with different representatives to determine the nodes of the set K that must be linked to graph the final simplified mesh. Due to the equivalence relation established in the original node set, associating each node n_i with the node k_j the set K closer to it, there are no triangles in the final mesh that intersect each other.

We have added a condition at the end of the triangulation algorithm. The primary objective of this condition is to avoid the existence of holes or unconnected regions in the final mesh. When the shapes of the original meshes are not too complicated, we have no problem in the final result with holes or isolated regions; however, as it has been observed in the examples performed, when the shape of the original mesh is really complex, we can have problems with the appearance of unconnected regions or isolated vertices, specially when $M \ll N$.

Another aspect that we want to highlight is related to the computational cost of the triangulation process. The only part that requires a cost of computational operations is point 1 of the algorithm, where we associate each node of the original mesh A with a node of the set K. We need to compute, for a fixed n_i,

$$d^2(n_i, k_1), d^2(n_i, k_2), \ldots, d^2(n_i, k_M),$$

which represents 8 arithmetic operations each time we compute the square distance. As we repeat these computations for every k_j, with $j = 1,2,\ldots,M$, the cost will be 8 M operations. In addition, as the number of nodes of A is N, we repeat this procedure N times, which represents a total cost of O (M N), with $M \ll N$. If we consider that the direct implementation of the Delaunay method leads to algorithms whose computational complexity is O (N^2), that is, quadratic, we can conclude that the computational cost of the triangulation proposed in this algorithm is efficient from the point of view of its computational complexity.

4. Examples and numerical experiments

To assess the performance of the algorithm proposed in Section 2, several experiments were conducted using some original 2D triangle meshes. Both the self-organizing algorithm and the triangulation process were implemented in Matlab. The numerical examples performed with 2D triangle meshes of different sizes and topologies show us, as a general characteristic, that our model can produce high fidelity approximations of any original triangle mesh, no matter how complex the shape of the mesh.

To carry out the experiments, it is necessary to specify a set of parameters to run the self-organizing algorithm. The parameters that must be specified are the following:

- ϵ_{win} is related to the displacement of the winner node.
- ϵ_n is related to the displacement of the neighbor nodes.
- λ was introduced in order to be sure that any node of the initial set of random points will stay isolated during the entire execution of the algorithm.

There is no parameter set that can always produce the best results (lowest error values) for all possible 2D triangle meshes. The parameter values that minimize error for a certain model can achieve slightly higher error values for other meshes. This characteristic is common in all training processes based on neural networks. Nevertheless, the simplified models present a very high similarity with the original meshes regardless of parameter values, since the error is still very small. Therefore, there is no theoretical result that provides us a parameter set that can always produce the best results for the simplified mesh. Consequently, the determination of the set of parameters to run the algorithm is obtained experimentally. The values of these parameters used in our examples have been $\epsilon_{win} = 0.8$, while $\epsilon_n = 0.1$. The parameter λ is not fixed, as it depends on the number of nodes N; in many examples we used $\lambda = 100 \cdot N$.

The first example we analyze is related to wireless networks. A wireless network can be represented as a graph in which the nodes represent wireless devices, and the links represent pairs of nodes that communicate directly by means of radio signals. In this example we consider a network of 16 nodes and 18 triangles. This network represents the old center of a real town with geographical coordinates (38.25, –0.7) (latitude and longitude). Each node of this network represents a public space of the town, like parks, gardens, municipal buildings, facilities, ... see Fig. 1 (left image) for a graphical representation of the original 2D network over the geographical area.

We want to build a wireless network capable of covering the entire old center area which is given by the original network. The simplified network should consist of 8 hubs, from which it distributes the wireless signal that must reach the critical points of the original network. In other words, the simplified network provides us the geometric positions where we need to place WiFi nodes to enable citizens to have connection in public areas. See [2] for a detailed description of some models to construct minimum-interference wireless networks.

The starting point of the self-organizing algorithm is to place 8 points, $k_1, k_2, \ldots, k_8$, at random positions in the geographical area. We take 8 points because it is the number of nodes that we want for the simplified network. Fig. 1 (right image) shows the location of the initial random nodes. Note that most of the points are outside the shape of the original network.

Remember that the term nodes or vertices refers to wireless devices. After randomly generating the initial positions of the 8 nodes of the simplified mesh, we continue running the training process in the self-organizing algorithm. In this case, after

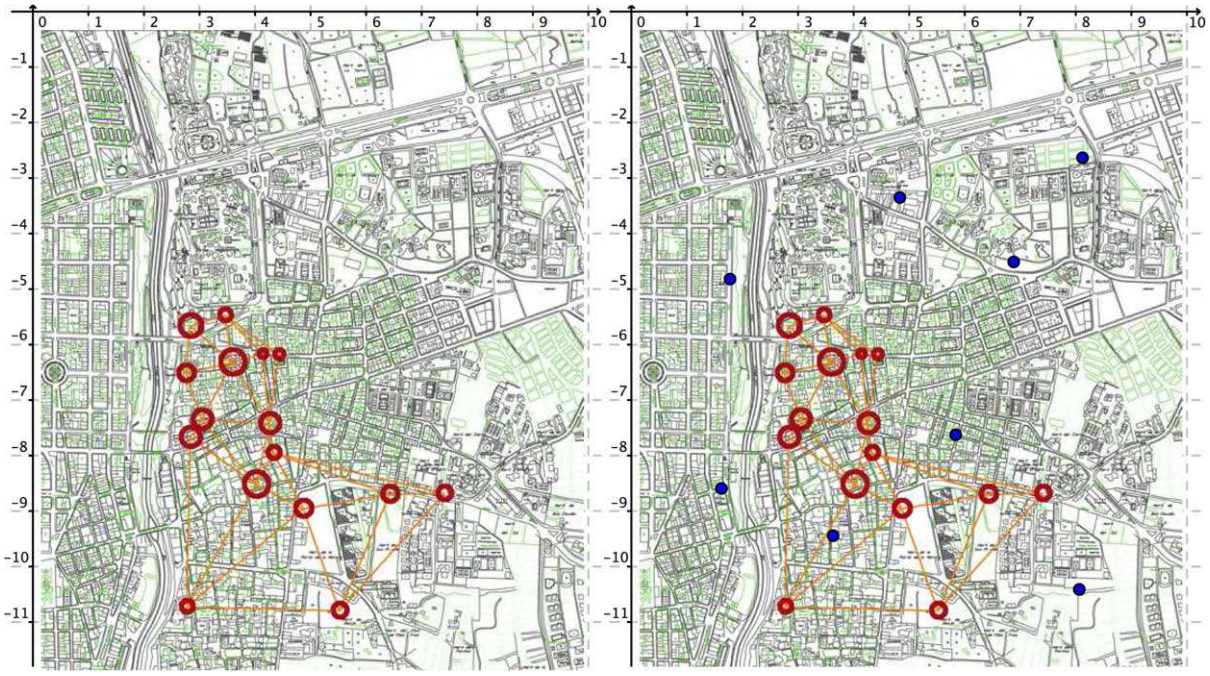

Fig. 1. The image on the left represents the original 2D triangle mesh, while the image on the right represents The original 2D triangle mesh with the initial 8 nodes placed at random positions.

150 iterations, we check the local counters of the $n_1, \ldots, n_{16}$ nodes of the original network; then, we proceed to remove the vertices that have not been winners in any iteration. After this step, we continue with the iterations.

After performing 10,000 iterations, we stop the process, obtaining the final positions of the vertices for the simplified network. The experimental result obtained is shown in Fig. 2.

It should be noted that all of the initial nodes have been properly moved to the area enclosed under the original network, even though some of the initial nodes were located far away from the grid. That is a proof of the model efficiency.

However, there is another aspect that we must also stress, as it is the fact that in the areas where there is a greater concentration of original vertices, there is a greater probability that the vertices of the final mesh are located. This is due to the training process of the self-organizing algorithm. And we want to highlight this feature because it constitutes an advantage over other methods.

At this moment it is possible to carry out the triangulation process from the information provided by the self-organizing algorithm. The basis of the triangulation process is the comparison between the original nodes of the network and the new nodes. The reconstruction of the simplified mesh is shown in Fig. 2.

This simple example shows the efficiency of the model, especially the part related to the self-organizing process; if we look at the figures with the initial nodes and the figure obtained after running the algorithm, and compare them, it is clearly observed how the initial points (randomly generated) move towards the original nodes and remain in the area covered by the original network. This may be interpreted as a self-organized process in which the shape of the original network is learned through successive iterations. Running the algorithm several times for this example, we found that it is not necessary to run many iterations to obtain good approximations of the original mesh.

The example of the wireless network is somewhat simple. The number of vertices is reduced and the shape of the network is not complex. Even if the algorithm is executed without taking into account the possibility of isolated vertices, the results are good and the initial vertices are moved to proper positions. Unfortunately, not all the examples present the same desired conditions.

In the collection of examples studied it has been observed that when the topology of the mesh is very irregular and the initial nodes are generated completely at random, it may be the case that when performing a low number of iterations, the geometric positions of the nodes are not optimal for the reconstruction of the final simplified mesh. In other words, when the above conditions are given, we suspect that some gaps or isolated nodes may appear in the simplification mesh. This behavior can be visualized in Fig. 3.

The behavior of the self-organizing algorithm shown in the mesh of Fig. 3 is not unusual and has a clear explanation. In the left image of Fig. 3, if we pay attention to the situation of the initial vertices, then it is noted that some of these points are far from the original network. When performing successive iterations, none of these nodes will be the winner in the training process; notice that the input signal is always a node of the original network. Therefore, they will never move towards the

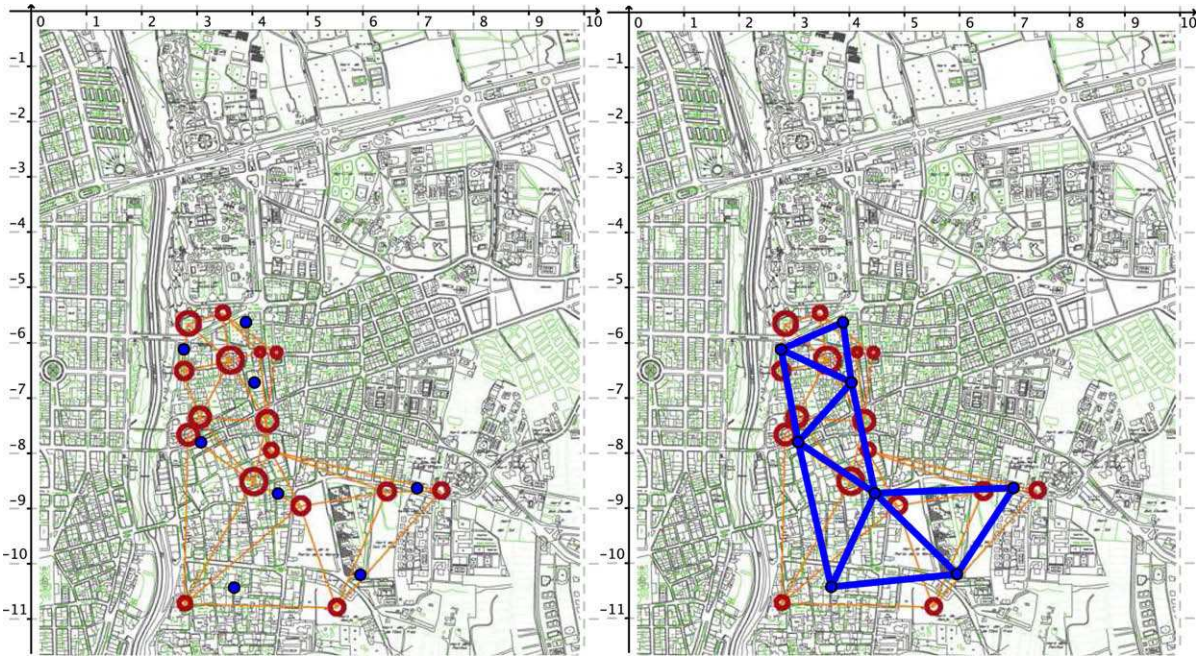

Fig. 2. The image on the left represents the original 2D triangle mesh with the final 8 nodes placed after running the self-organizing algorithm, while the image on the right represents The 2D simplified triangle mesh.

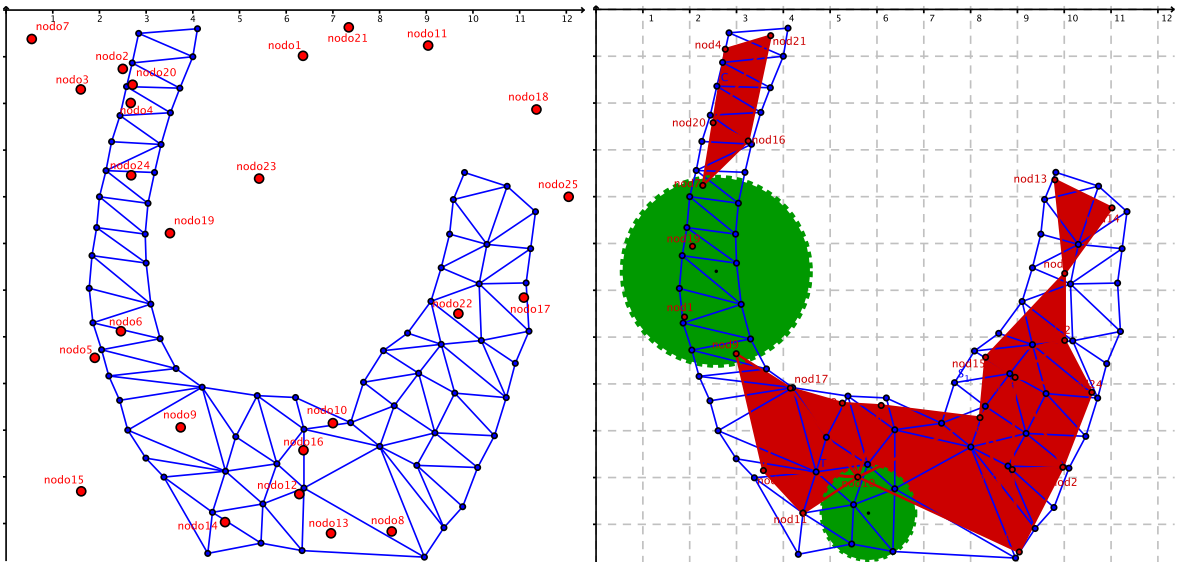

Fig. 3. The image on the left represents the original 2D triangle mesh with the 25 nodes placed at random, while the image on the right represents the final mesh after running the self-organizing algorithm and the triangulation process.

original mesh and the consequence of this is that this node is isolated and stays in the same position, regardless of the number of iterations.

The occurrence of these isolated vertices leads us to introduce the concept of local counter in the algorithm. It is easy to understand that the local counter of an isolated vertex remains zero. When the local counter is checked, really what we are doing is detecting the isolated vertices. This is the reason for introducing the point 7 in the algorithm. An example is shown in order to visualize this behavior.

In Fig. 3 we show an original network that presents a very irregular shape. In this example, the original network has 73 nodes and we want the number of nodes of the simplified network to be a third part of the number of nodes of the original mesh. That is, we have $N = 73$ and $M = 25$. Therefore, we have generated 25 nodes as a starting point in the self-organizing algorithm. The way in which the nodes have been generated has been absolutely at random. That means that no isolated node has been removed after a number of iterations, (remember that this is a basic step in the self-organizing algorithm). Moreover, we have run a very low number of iterations (2000).

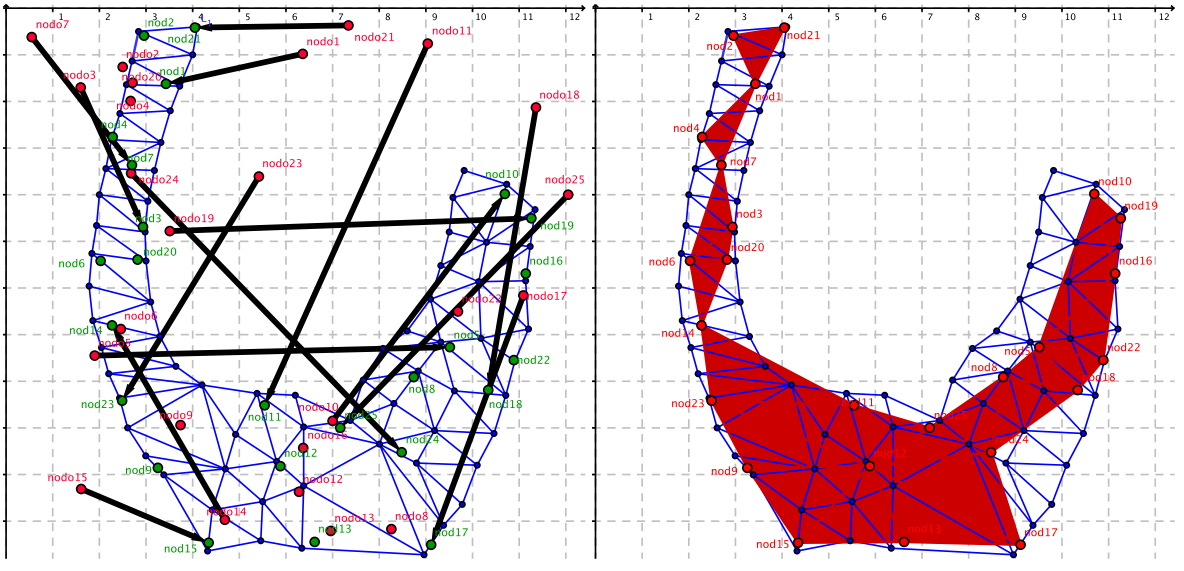

Fig. 4. The image on the left represents the displacement of some nodes, while the image on the right represents the final simplified mesh after running the self-organizing algorithm and the triangulation process, for 25 nodes.

In short, the most unfavorable conditions for the simplification of a two-dimensional mesh are given: a very irregular shape, a drastic reduction in the number of nodes and a very low number of iterations. The result is shown in Fig. 4, in which the isolated nodes and gaps in the simplified network are clearly visible. In any case, it is surprising how the simplified mesh perfectly preserves the topology of the original mesh.

We would like to emphasize the power of the self-organizing algorithm in terms of its ability to learn the shape of a given mesh or network, starting from a cloud of points generated at random. In all the experiments with different topology networks, the algorithm had no problem to place the initial nodes (placed at random) inside the area covered by the original network. We have demonstrated this fact with the example shown in Fig. 4. It is remarkable that none of the points generated absolutely at random are placed outside of the network (see the right image). The problem arises when the mesh must be reconstructed, because the new vertices are not in the best positions to carry out the triangulation. This produces the isolated vertices we can see in the shaded area of the right image in Fig. 3.

We can analyze in detail the different displacements experienced by the initial cloud of points (initial nodes) when running the algorithm, for this particular example. The most significant displacements of some final nodes with respect to its geometric position at the beginning of the algorithm, when they are generated at random positions, are illustrated by means

Table 1
Most significant displacement of nodes in Fig. 4 (left).

Node	Random x	Random y	Final position x	Final position y
1	6.36	−0.98	3.43	−1.62
3	1.60	−1.70	2.94	−4.69
5	1.90	−7.45	9.51	−7.27
7	0.55	−0.62	2.70	−3.36
10	7.00	−8.85	10.69	−3.98
11	9.04	−0.76	5.54	−8.52
14	4.69	−10.97	2.27	−6.80
15	1.61	−10.31	4.34	−11.46
17	11.09	−6.16	9.11	−11.51
18	11.36	−2.10	10.34	−8.19
19	3.51	−4.78	11.26	−4.51
21	7.34	−0.37	4.05	−0.42
23	5.42	−3.61	2.49	−8.42
25	12.05	−4.00	7.16	−9.00

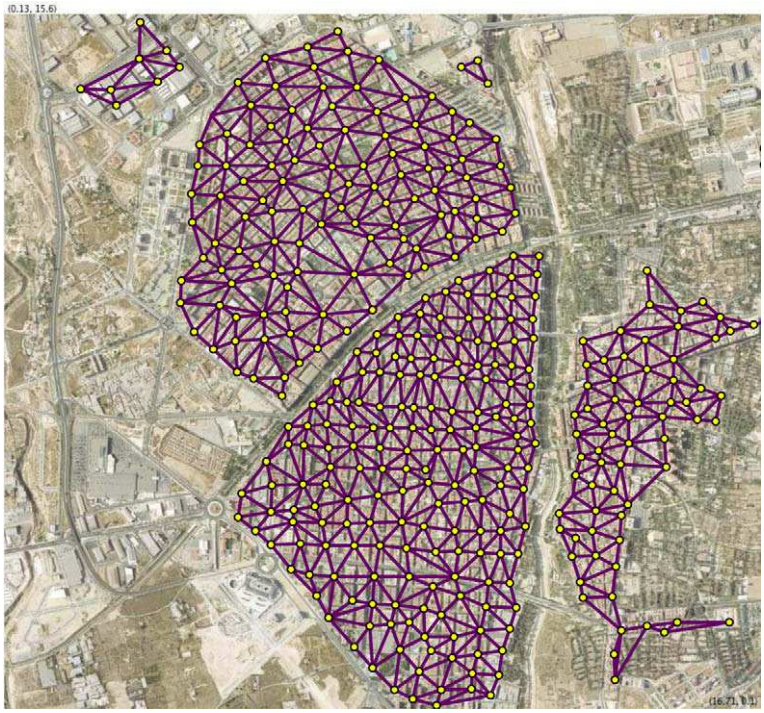

Fig. 5. A 2D triangle mesh covering the downtown of a city (Elche, Spain).

of Fig. 4 (left image). The numerical values for the displacements are summarized in Table 1. If we look at the table in detail, we note that there are some nodes that suffer large displacements of position. Especially significant is the displacement experienced by nodes labelled as 11, 19, 23, 24 and 25. For example, node 11 is initially placed at position (9.04, –0.76); it suffers a very important displacement to reach the position (5.54, –8.52).

The nodes outside the area of the mesh share the same feature: they move inside the mesh (see the displacements of nodes 1, 11, 15, 18, 19, 21, 23 and 25). The nodes inside the area of the mesh present, as a general characteristic, small displacements.

The problem related to isolated nodes or areas in the generation of the simplified mesh encourages the point 7 in the self-organizing algorithm. When we run the algorithm as it is presented exactly in Section 2, a final mesh which has no holes or isolated nodes is generated, as shown in Fig. 4 (right). The gaps appearing in the mesh generated in Fig. 3 disappear in the new mesh. The simplified network given by Fig. 4 was obtained after 100,000 iterations. We conducted some experiments

Table 2
Final position of the nodes in the mesh shown in Fig. 6.

Final nodes		Final nodes	
Node	Position	Node	Position
k_1	(9.91,10.74)	k_2	(11.19,2.06)
k_3	(15.57,8.47)	k_4	(10.78,4.26)
k_5	(7.42,8.27)	k_6	(13.50,2.71)
k_7	(9.35,8.53)	k_8	(11.16,9.22)
k_9	(8.29,13.47)	k_{10}	(6.75,11.42)
k_{11}	(6.16,7.03)	k_{12}	(8.17,5.11)
k_{13}	(4.72,13.70)	k_{14}	(5.42,8.64)
k_{15}	(3.13,14.97)	k_{16}	(9.76,1.41)
k_{17}	(13.25,8.33)	k_{18}	(6.36,4.61)
k_{19}	(10.28,7.47)	k_{20}	(14.59,1.83)
k_{21}	(9.00,3.21)	k_{22}	(15.75,6.88)
k_{23}	(8.78,9.97)	k_{24}	(11.32,6.28)
k_{25}	(13.01,5.82)	k_{26}	(10.06,12.12)
k_{27}	(4.30,12.44)	k_{28}	(8.19,6.82)
k_{29}	(4.52,9.82)	k_{30}	(7.61,2.60)

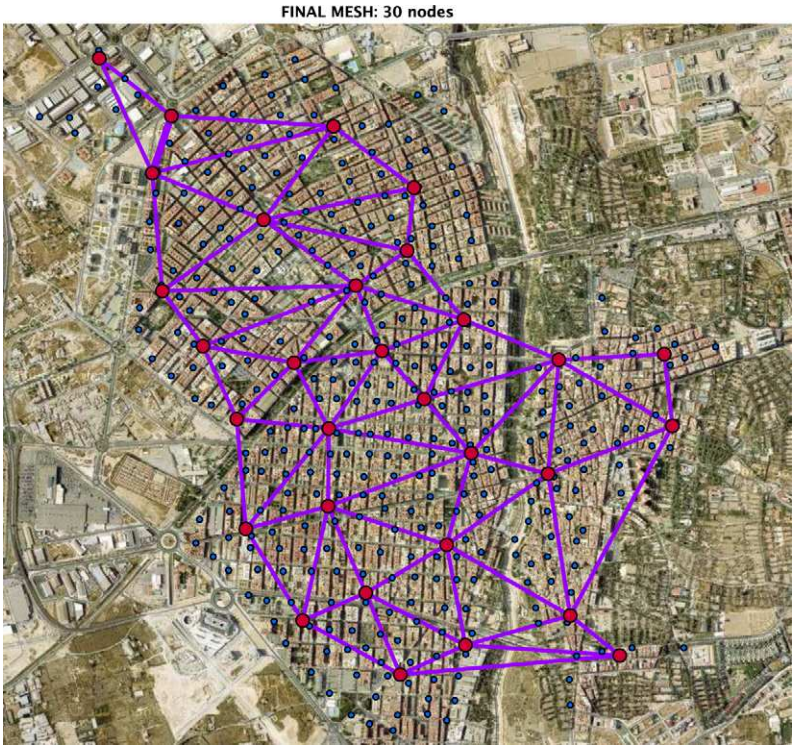

Fig. 6. The final simplified mesh covering the downtown of the city.

varying the number of iterations and the results were equally satisfactory. So, for 75,000 and 150,000 iterations, the simplified networks were quite similar to that shown in the figure. No gaps were generated in the triangulation process.

The self-organizing characteristics of the algorithm in the model have been clearly revealed in the above example. It is remarkable that even the most distant nodes from the area covered by the original mesh suffer a considerable displacement to match the shape of the mesh. Therefore, we do not mind that the shape of the original network is irregular as this capacity for self adjustment of the algorithm causes no unexpected results to occur. In all of the original 2D meshes studied with different sizes and shapes, the results have been equally successful in the sense that we simplify the original mesh, preserving the topology of the initial network. No matter how irregular the shape of the original mesh is; the self-organizing algorithm always *learns* how the shape is and places the initial nodes in the original mesh.

To finish this section we show a new example of simplified 2D meshes in which the number of vertices increases considerably and, therefore, the complexity of the mesh. We consider the problem of designing a Wifi network in the downtown of a city consisting of 30 nodes or hubs, with the primary objective that the signal covers the entire urban area. To carry out this problem, we construct a 2D triangles mesh, where each node represents a block house or housing blocks and perform a triangulation with the nodes drawn. We see the original mesh in Fig. 5.

This initial mesh in Fig. 5 has 367 vertices or nodes (houses) and 950 edges. Now, the objective is, by means of the self-organizing algorithm described in Section 2, determine a set of positions in the urban area in which to place the hubs or nodes that will distribute the Wifi signal to everywhere in the urban area. Let us assume, therefore, that we want to obtain a simplified mesh with 30 nodes and, what is more important, we need that this reduced grid looks like as possible to the original mesh. We run the self-organizing algorithm starting with 30 nodes created at random. We remove the isolated nodes after $\lambda =$ 50,000 iterations. After $I_t = 10^9$ iterations, the resulting nodes are shown in Table 2.

Once we have obtained the points for the final mesh, it is possible to carry out the triangulation process from the information provided by the self-organizing algorithm. The basis of triangulation process is the comparison between the original nodes of the network and the new nodes. The reconstruction of the simplified mesh is shown in Fig. 6. Notice how the simplified mesh has been able to *learn* the topology of the original mesh.

5. Conclusion

This paper has introduced a new 2D triangle mesh simplification model with the property of preserving the shape of the original mesh. The model presented consists of a self-organizing algorithm whose objective is to generate the positions of the nodes of the simplified mesh; afterwards, a triangulation algorithm is carried out to reconstruct the triangles of the new simplified mesh. The model seeks to minimize the number of parameters that control the training process.

The experimental results obtained using some initial meshes with different shapes and complexities, show that the final simplified meshes are generated efficiently and with a high level of approximation with respect to the original meshes, no matter how irregular shapes they present. Even for the most irregular networks, the approximations obtained fit perfectly to the original shape, as demonstrated in the examples presented.

References

[1] R. Alvarez, J. Noguera, L. Tortosa, A. Zamora, A mesh optimization algorithm based on neural networks, Information Sciences 177 (2007) 5347–5364.
[2] M. Benkert, J. Gudmundsson, H. Havenkort, A. Wolff, Constructing minimum-interference networks, Computational Geometry 40 (2008) 179–194.
[3] S. Bereg, Transforming pseudo-triangulations, Information Processing Letters 90 (3) (2004) 141–145.
[4] P. Bose, F. Hurtado, Flips in planar graphs, Computational Geometry: Theory and Applications 42 (1) (2009) 60–80.
[5] P. Castello, M. Sbert, M. Chover, M. Feixas, Viewpoint-based simplification using f-divergences, Information Sciences 178 (11) (2008) 2375–2388.
[6] P. Cignoni, C. Montani, R. Scopigno, A comparison of mesh simplification algorithms, Computer Graphics 22 (1) (1998) 37–54.
[7] B. Delaunay, Sur la sphere vide, Bulletin of Academic of Science USSR VII: Classe de Scienze Matematiche Naturali (1934) 793–800.
[8] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi tessellations: applications and algorithms, SIAM Review 41 (1999) 637–676.
[9] M. Held, J.S.B. Mitchell, Triangulating input-constrained planar point sets, Information Processing Letters 109 (2008) 54–56.
[10] L. Ju, Q. Du, M. Gunzburger, Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations, Parallel Computing 28 (2002) 1477–1500.
[11] T. Kohonen, Self-organizing formation of topologically correct feature maps, Biological Cybernetics 43 (1982) 59–69.
[12] D.P. Luebke, A developer's survey of polygonal simplification algorithms, IEEE Computer Graphics and Applications 21 (3) (2001) 24–35.
[13] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: L. Le Cam, J. Neyman (Eds.), Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. I, University of California, 1967, pp. 281–297.
[14] H. Nguyen, J. Bunkardt, M. Gunzburger, L. Ju, Y. Saka, Constrained CVT meshes and a comparison of triangular mesh generators, Computational Geometry 42 (1) (2009) 1–19.
[15] J. Noguera, L. Tortosa, A. Zamora, Analysis and efficiency of the GNG3D algorithm for mesh simplification, Applied Mathematics and Computation 197 (2008) 29–40.
[16] M. Sharir, E. Welzl, Random triangulations of planar point sets, in: Proceedings of the 22nd Annual Symposium on Computational geometry, 2006, pp. 273–281.
[17] F. Vivodtzev, G.-P. Bonneau, P. Le Texier, Topology preserving simplification of 2D non-manifold meshes with embedded structures, Visual Computation 21 (8) (2005) 679–688.
[18] J. Yu, Surface reconstruction from unorganized points using self-organizing neural networks, in: Proceedings of IEEE Visualization'99, San Francisco, CA, 1999, pp. 61–64.