Neural Comput & Applic (2006) 15: 366–372
DOI 10.1007/s00521-006-0041-2

ORIGINAL ARTICLE

C. S. Chong · H. P. Lee · A. Senthil Kumar

Genetic algorithms in mesh optimization for visualization and finite element models

Received: 25 October 2005 / Accepted: 17 February 2006 / Published online: 4 April 2006

Abstract This paper investigates the use of a genetic algorithm (GA) to perform the large-scale triangular mesh optimization process. This optimization process consists of a combination of mesh reduction and mesh smoothing that will not only improve the speed for the computation of a 3D graphical or finite element model, but also improve the quality of its mesh. The GA is developed and implemented to replace the original mesh with a re-triangulation process. The GA features optimized initial population, constrained crossover operator, constrained mutation operator and multi-objective fitness evaluation function. While retaining features is important to both visualization models and finite element models, this algorithm also optimizes the shape of the triangular elements, improves the smoothness of the mesh and performs mesh reduction based on the needs of the user.

Keywords Genetic algorithms · Mesh optimization · Finite element modeling

1 Introduction

There are two main motivations in the work presented in this paper:

1. To improve the mesh quality of a finite element model with poorly shaped elements and to reduce computation time by cutting down the number of elements of the original finite element model that are finely meshed
2. To handle and reduce complex 3D models which are difficult to render fast due to the large number of triangles present

The goal of most finite element analyses (FEA) is to verify the suitability of an engineering design. The challenge is to build a sufficiently accurate model in the available time. One of the most time-consuming tasks in building a finite element model is generating and optimizing the finite element mesh. The number of triangles or elements can be further increased or decreased depending on the application requirement as long as the data in these meshes would not affect the accuracy of the simulation processes.

In addition, real-time graphics are becoming increasingly prevalent in our world. Computer games, training simulations and medical imaging all rely on interactive graphics. For the most part, complex geometric models comprising meshes of triangles are the backbone of such systems. When digitizing a part, in order not to miss any detail of its geometry, a large number of measurement points are normally collected. Such models allow us to display arbitrary model geometry in real time, but there is a significant rendering cost in drawing all those triangles. Reducing the number of triangles in our models would allow us to render scenes faster, and bigger and more complex scenes interactively. In fact, keeping lots of data points in planar or nearly planar region is rather unsophisticated.

As a result of their global optimization property [1, 2], genetic algorithms (GAs) have been widely used in various fields such as state space search, nonlinear optimization, machine learning, traveling salesman problems, etc. [3]. Both theoretical [4, 5] and experimental studies [6, 7] show that the genetic technique is an efficient and robust heuristic for search in complex spaces solving complex optimization and combinatory optimization problems.

In recent years, preliminary study on GAs in triangulation has been reported. Absaloms and Tomikawa [8] propose a GA to triangulate two adjacent contour data

C. S. Chong (✉) · H. P. Lee
Institute of High Performance Computing,
1 Science Park Road #01-01, The Capricorn,
Singapore Science Park II, Singapore,
Singapore 117528
E-mail: chongcs@ihpc.a-star.edu.sg

A. Senthil Kumar · H. P. Lee
Department of Mechanical Engineering, National
University of Singapore, Singapore, Singapore 119260

from a digitized geographical map. They claim that GA-based triangulation is relatively technique independent and can be implemented by parallel processing. A GA-based method for simple 2D triangulation is also reported recently [9]. In the report, the GA-based triangulation is compared to the result from greedy triangulation. It is concluded that the GA-based method will lead to better optimization results.

Genetic algorithms were first developed by John Holland at the University of Michigan in the mid-1970s and are the subject of much research today. GAs are robust and, although they may not find a perfect solution, they come close enough for most engineering work for a wide variety of problems. Multimodal and highly discontinuous problems are taken in stride by GAs. There are numerous variations on GAs including different types of crossover algorithms, hybrids combining GAs with fuzzy logic, simulated annealing and neural networks. Hamann [10] proposes a data reduction scheme for the triangulated surface. He removes a triangle based on the curvatures at the three vertices. A user can specify a percentage of triangles to be removed. His research has smooth surface fitting in mind. Hoppe et al. [11] use an energy function to represent the trade-off between geometric fit and compact representation. A user-desired parameter is used to control the trade-off between geometric fit and compact representation. A large value indicates that a sparse representation is strongly preferred, but at the expense of degrading the fit. A merging algorithm based on edge collapse is proposed for automatically computing approximations of a given triangulated object at different levels of detail [12]. Edges are queued according to their cost functions, which indicate the error caused by edge collapse. Approximation levels are controlled by prescribing geometric tolerances. Gieng et al. [13] classify mesh simplification algorithms into three types: removing vertices, removing edges and removing faces.

2 Methodology

In this paper, attempts are made to create a mesh optimization system using genetic algorithms that would allow the user to create low-detail models interactively without significant loss of mesh fidelity. Figure 3 shows the workflow of the entire optimization process. In this paper, we deal with 2D triangular elements as 3D visualization models are made up of triangles and triangles are the basic elements in finite element modeling. The first step of the optimization process is to remove patches of triangular elements that form empty regions which will be "re-meshed" and optimized subsequently based on a genetic algorithm. There are two methods of removing triangular mesh patches as described in Sect. 2.1. These two methods can be executed independently or together depending on whether the user just wants to improve the mesh quality or reduce the mesh size of a model or perform both, depending on the user's requirements. Often in 3D visualization models, the quality of the triangles is not an issue as compared to the smoothness and features of the meshed model and the number of triangles present in it. However, for finite element models, the shape of the triangle is vital for finite element simulation. Mesh reduction coupled with mesh smoothing is good if the user wants to coarsen a very finely meshed model for faster computation.

2.1 Removal of triangles

In this paper, two methods of removing triangles are presented as follows:

(a) *Removal of triangles associated to the nodes of an element*: This algorithm for the removal of elements for subsequent mesh re-creation is designed for reducing the number of triangular elements of a triangulated model to speed up processing time. Here we are trying to remove a group of connected elements that are adjacent to the nodes of a specified element. Figure 1 shows the surrounding elements, B1–B11, associated with the three nodes belonging to the affected element, A. Element A along with elements B1–B11 face possible removal. A normals' deviation factor will decides the removal of elements. The calculation of the normals' deviation factor for element removal is as follows:

1. First the deviations of the unit normals of the surfaces of the surrounding elements from the chosen element are calculated using (1) and (2), where A is the unit normal of the chosen element, B_i are the unit normals of n number of affected elements and d_i is the deviation of the unit normal of B_i from A with $i = 1,2,3,\ldots, n$.
2. Next, the average deviation is calculated using (3) and the final deviation factor will consist of the total value of the average deviation and the difference in the maximum and minimum values of d_i. The smaller the value of d, the higher the probability that the chosen elements and its affected surrounding elements will be removed. The deviation factor will rank in a hierarchical manner and the user can specify a tolerable deviation value or a percentage of elements with the lowest deviation factor and usually start off with the element having the lowest deviation factor:

$$AB_i = B_i - A = x_i i + y_i j + z_i k, \tag{1}$$

$$d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}, \tag{2}$$

$$d_{\text{avg}} = \frac{\Sigma d_i}{n}, \tag{3}$$

$$d = d_{\text{avg}} + d_{(\max - \min)}. \tag{4}$$

(b) *Removal of triangles associated to the edges of an element*: This method for the removal of elements for subsequent mesh re-creation is designed for mesh

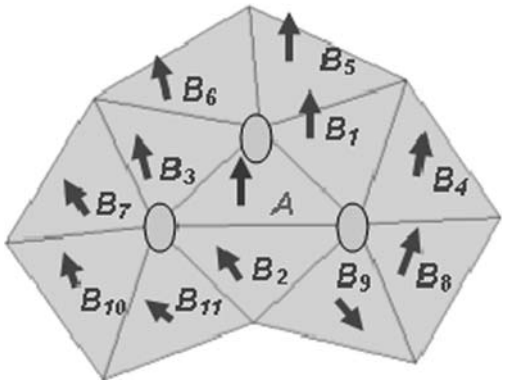

Fig. 1 Removal of triangles associated to the nodes of an element

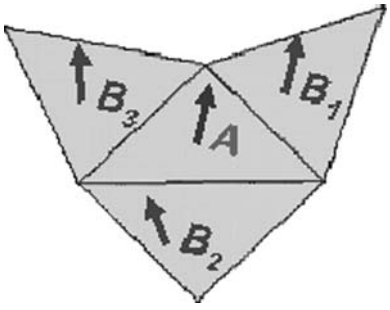

Fig. 2 Removal of triangles associated to the edges of an element

smoothing, in order to achieve quality mesh that is critical for accurate FEA. The number of elements of the mesh will not be reduced. A group of connected elements which are adjacent to the elements of a specified element will be selected to be removed. Figure 2 shows the surrounding elements, namely, B1, B2 and B3, together with the affected element A, facing possible removal. The deviation factor that decides the sequence and the elements for removal is the same as the above using (1)–(4).

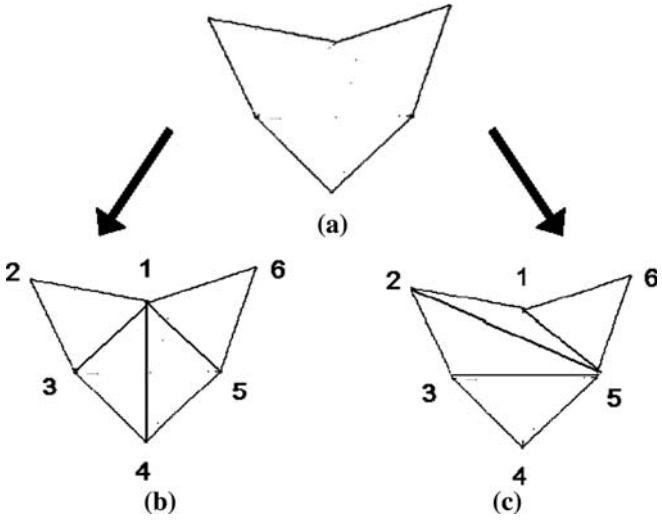

Fig. 3 **a** Re-triangulation of an empty region, **b** {[1–3], [1–4], [1–5]} links and **c** {[1–5], [2, 5], [3–5]] links

2.2 Re-triangulation using genetic algorithm

The re-triangulation takes place when a region of elements is being removed. Re-triangulation links all points on the boundary of the empty region to form triangular facets(elements). Figure 3 shows an example of re-triangulating an empty region caused by the removal of elements associated to the three edges of a selected element with a low normals' deviation factor. From Table 1, we can calculate the total number of possible links and their combination. This table can be computed based on the number of boundary edges or points (nodes) present in the empty region using (5):

$$\text{Total number of possible links}: \quad C = (n-3) + \sum_{i=1}^{n-3} i, \tag{5}$$

where n is the number of boundary edges present in the empty region.

Equation (5) can be further simplified to

$$C = \frac{1}{2} n(n-3). \tag{6}$$

The possible links are derived as follows:

1. Set any boundary point as the first node and label the rest of the points in an anti-clockwise (or clockwise) manner as in Fig. 3b.
2. Assign number of possible links to each boundary points starting from the first node. Nodes 1 and 2 will always have $(n-3)$ links and the subsequent nodes will have $(n-4)$ links, $(n-5)$ links, and so on, and the last two nodes will always have 0 link. In this way, no duplicated links will be present.
3. Next we will describe each link starting from node 1. In Fig. 3, there are six boundary edges present in the empty region. The three links of node 1 will be [1–3], [1–4], [1–5] as the [1–2] link will be ignored as it is the boundary edge. Similarly for node 2, the [2–3] link will be ignored and the three possible links are [2–4], [2–5], [2–6]. The possible links for node 3 will be [3–5], [3–6] and node 4 will be [4–6]. The last two nodes will not have any link.
4. Check for any invalid links among the possible links. For example in Fig. 3, [2–6] is an invalid link and will be discarded.
5. So the valid links as in Fig. 3 are {[1–3], [1–4], [1–5], [2–4], [2–5], [3–5], [3–6], [4–6]}.

Thus in the chromosome representation, eight binary characters or bits represent the eight links. The chromosomes will be like 11100000 as in Fig. 3b. There will

Table 1 Construction of links between nodes of a hole (empty region) boundary

	No. of free edges in the hole (n)							
	3	4	5	6	7	8	⋯	n
No. of links starting from								
1st node	0	1	2	3	4	5	⋯	$n-3$
2nd node	0	1	2	3	4	5	⋯	$n-3$
3rd node	0	0	1	2	3	4	⋯	$n-4$
4th node	NA	0	0	1	2	3	⋯	$n-5$
5th node	NA	NA	0	0	1	2	⋯	$n-6$
6th node	NA	NA	NA	0	0	1	⋯	⋯
7th node	NA	NA	NA	NA	0	0	⋯	⋯
8th node	NA	NA	NA	NA	NA	0	⋯	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$n-3$th node	NA	NA	NA	NA	NA	NA	1	2
$n-2$th node	NA	NA	NA	NA	NA	NA	NA	1
$n-1$th node	NA	NA	NA	NA	NA	NA	NA	0
nth node	NA	NA	NA	NA	NA	NA	NA	0
Total no. of possible links (C)	0	2	5	9	14	20	⋯	$(n-3)+\sum_{i}^{n-3} i$

be checks to ensure that the new triangles will not overlap one another, i.e., no crossings of the links. At any one time during the crossover process, there will only be three 1s. It can be seen from Fig. 3 that no matter how the crossover takes place, the resulting triangulation always consist of four triangles which is the same number as the removed triangle. Thus, the method of removing elements that are associated with the three edges of an element will not result in any change in the number of element but will improve the quality of the mesh during the crossover process based on a fitness test. However, the removal of elements associated to the three nodes of a selected element may result in the formation of an empty region with anything equal to or more than three boundary edges. This often results in a reduction of elements when we use the genetic algorithm for re-triangulation of the empty regions.

Figure 4 is a flow chart of the GA and the solution method taken in this approach to the smoothing process. The genetic algorithm then works as follows:

1. The initial population is filled with chromosomes that are generally created at random. Each chromosome in the current population is evaluated using the fitness measure. The more fit solutions reproduce and the less fit solutions die off. The fitness value is computed taking into consideration the shape of the triangles to be created, the smoothness of the triangles to be created and the smoothness of the triangulated patch with its surrounding triangles. The shape factor will tell us how "equilateral" the triangle is. Each of the triangles created is represented by i from 1 to $n-2$. Every re-triangulation of an empty region due to the removal of element associated to the selected element's nodes will reduce the total number of elements in the model by 2. θ_{ij} denotes the three angles of each of the triangles created (where $j = 1, 2$ and 3). The larger the value of f_a the better the overall quality of the new triangles created. The calculation of the shape factor is shown in (7). The overall smoothness of the triangles created is calculated at the links where the angles between the normals of two adjacent triangles are calculated. A small angle will have large value of f_b which denotes smoothness in (8). Similarly for f_c in (9), we calculate the angles between the normals of the newly created triangles with the surrounding original triangles. The total factor is the summation of all the sub-factors as shown in (10), where

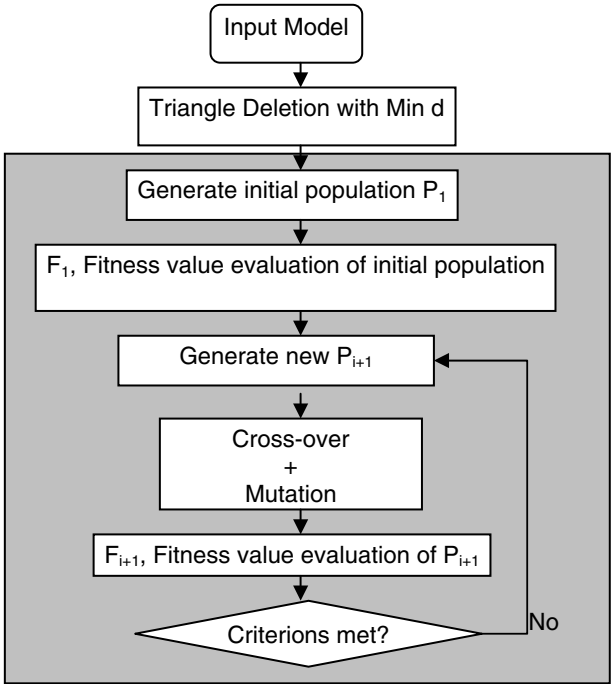

Fig. 4 Work flow of the mesh optimization process

each factor is normalized and multiplied by a weighing factor w, where $w_a + w_b + w_c = 1$. In the case of reducing the mesh of a 3D graphical model where the triangles' shapes are not critical, the weighing factor of the shape fitness value w_a can set to a lower value. In our implementation for models when the quality of the triangles is important, we set $w_a = w_b = w_c = 1/3$:

$$\text{Shape fitness factor, } f_a = \sum_{i=1}^{n-2}\left[\frac{2}{3}\pi^2 - \sum_{j=1}^{3}\left(\theta_{ij} - \frac{\pi}{3}\right)^2\right]. \tag{7}$$

Smoothness factor of the created triangles,

$$f_b = \sum_{i=1}^{n-3}(\pi - \alpha_i)^2, \tag{8}$$

where α_i is the angle between the normals of two adjacent elements to be created.

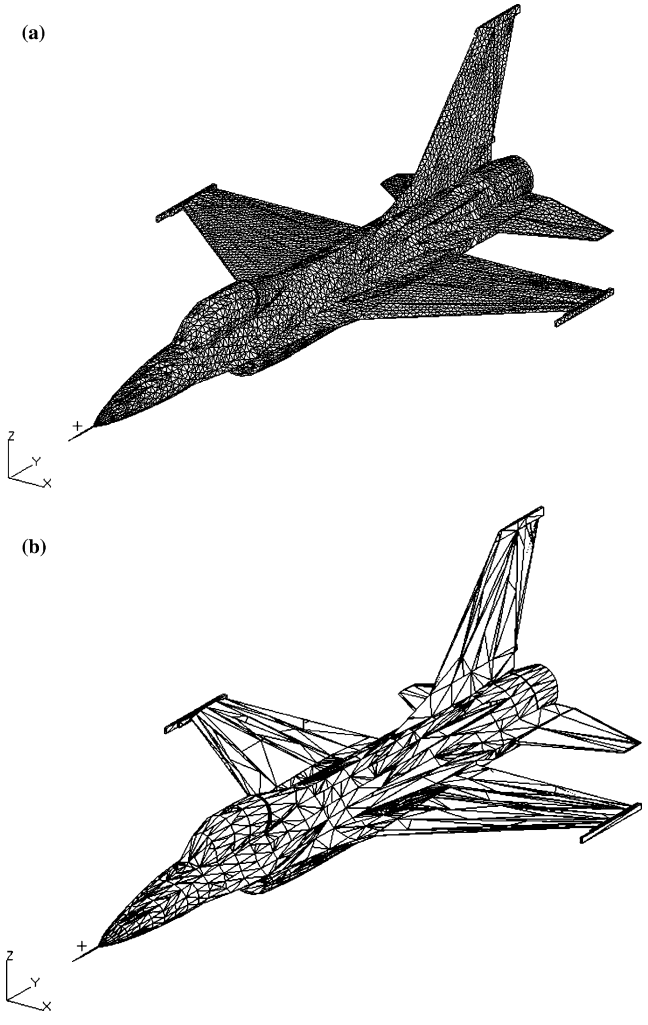

Fig. 5 **a** A triangular mesh of an aircraft with 32,186 elements and **b** the optimized mesh of the aircraft with 3,709 elements

Smoothness factor of the triangulated patch with its surrounding triangles,

$$f_c = \sum_{i=1}^{n}(\pi - \beta_i)^2, \tag{9}$$

where β_i is the angle between the normals of the created and the unaffected surrounding elements.

Total fitness value,

$$f_{\text{fit}} = w_a \frac{1}{(n-2)\pi^2} f_a + w_b \frac{1}{(n-3)\pi^2} f_b + w_c \frac{1}{n\pi^2} f_c, \tag{10}$$

where $w_a + w_b + w_c = 1$.

2. If the termination criterion is met, the best solution is returned. In our implementation in models where the quality of triangles is significantly important, we set $w_a = w_b = w_c = 1/3$. In the case when the triangles' quality is of no significance with pure triangles reduction and feature preservation, we set $w_a = 0$ and $w_b = w_c = 1/2$. The GA iterates until a terminating condition is met. In our implementation, we set the maximum number of iterations to be 500 for each GA computation and the best solution set is taken as the final re-triangulation of the affected empty region. Moreover, we allow the user to specify a desired total fitness value ($f_{\text{fit}} = (0,1]$) so that the algorithm terminates when the value is reached. A good empirical value of f_{fit} is 0.7.

3. Actions starting from step 2 are repeated until the termination criterion is satisfied. Crossover is used on two mates, producing two children. Elitism selection is used to copy the best 30% of the strings to the next population, which remain unaffected by crossover and mutation. The crossover probabilities are set to 0.70. Uniform crossover was used, which exchanged 30% of the bits between the mated strings.

A normal point mutation was used. The mutation probabilities can be as high as 0.5 as it is basically a random walk in the search space; perform edge swapping in the empty region. The convergence process may be a good application for fuzzy logic. We are not so lucky as to have a simple "go" or "no go" situation. We have a "maybe" or "maybe not" which is a natural for fuzzy logic. Given the way the GA works in this application, there will be many instances where the chromosome is considered totally "unfit" or invalid and the solution is discarded. This occurs when the links are crossed, i.e., two interior links intersect each other. Therefore, a gradual relaxation of the convergence criteria is required. This involves an increase in iteration number followed by lowering the total evaluation fitness value.

Once an empty region is re-triangulated and improved as much as possible, the GA moves on to the next worst region. This process of moving to the worst area of the model continues until a global model minimum has been reached. Here a global minimum considers the normal deviations for all elements in the

model. This global minimum is reached when the lowest normals' deviation factor exceeds the normal deviation tolerance input by the user or the number of maximum re-triangulations exceeds the user's input.

3 Case studies

Figure 5 shows an example which demonstrates the capability of the developed GA mesh optimizer, where a complex finite element model of an aircraft in Fig. 5a, containing 32,186 elements, is to undergo mesh reduction for faster reduction and manipulation in a 3D visual environment. Less emphasis is placed on the shape of the triangles but the features of the model will be retained. Figure 5b shows the outcome of the algorithm where this model is being reduced to only 3,709 elements, almost 90% reduction in the number of elements. Figure 6 shows a case study of a model of a heart. Figure 6a shows the original polygonal (mesh) model of the heart with 7,170 elements. Figure 6b shows the model being reduced almost 60% in the number of elements. This time, the fitness's weights associated with the triangles' quality in the mesh are set higher. This model of the heart contains very little "almost planar" regions and thus the reduction rate is lower if we were to preserve the features of the model, unlike the aircraft model which contains quite a fair bit of planar regions.

4 Drawbacks

The main drawback in using GA is the speed of the mesh optimization process. The optimization process handles the examples in Figs. 5 and 6 for 1 h 49 min and 53 min, respectively, on a P3-800 MHz PC, using C++. These types of timings are far from being called "real time". The main weakness of this algorithm is the slow computational speed even with high-performance workstations. Research on parallelization will be studied in the hope to shorten the loading as well as the computational time.

5 Conclusion

Although this work using genetic algorithm is yet to be conclusive, it does provide some promising results. Though clearly not the most efficient algorithm due to the time taken to run the optimization, the flexibility of the genetic algorithm makes it a potentially useful system for handling model containing huge data or large number of elements, for example, in the visualization of 3D models in graphical systems with data-size constraint, such as the CAVE system (The Electronic Visualization Laboratory, http://www.evl.uic.edu.sg). The domain-independent genetic algorithms make them perfect solutions in situations where the factors affecting the mesh fidelity are not fully understood or not easily enumerated. With a little work and some more experimentation, this work may become a practical alternative to the existing level of detail systems.

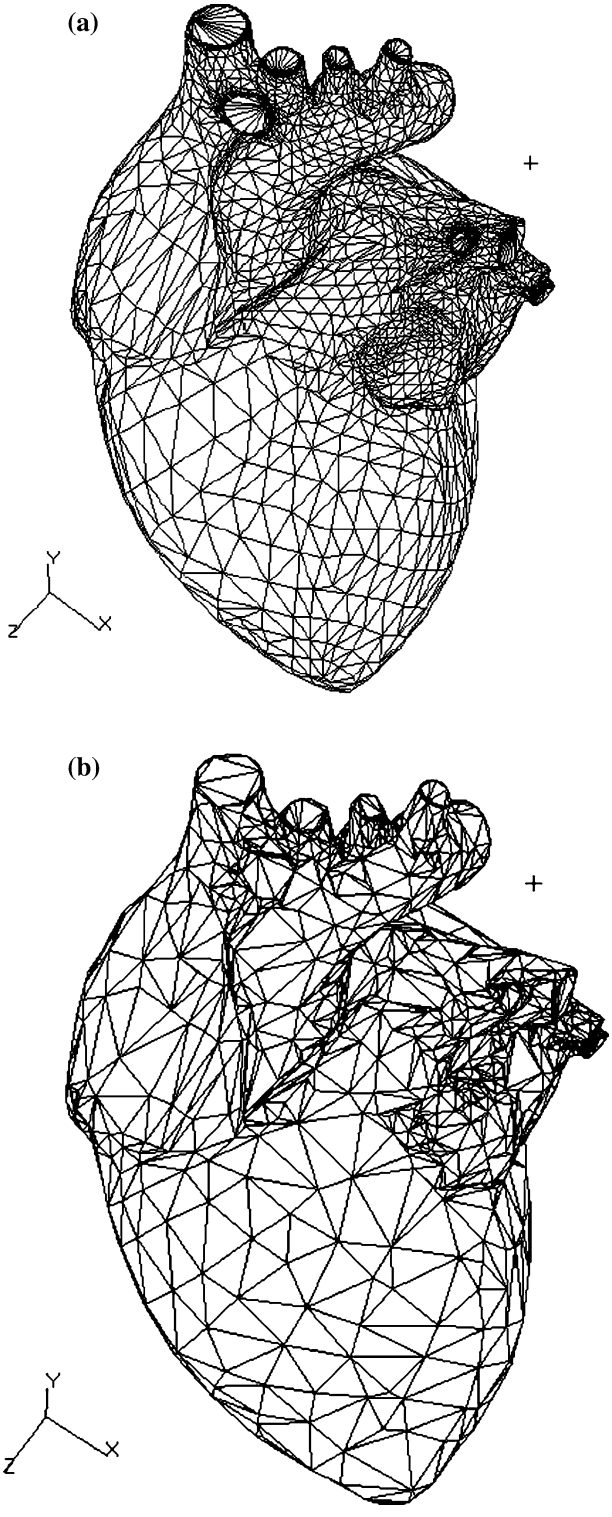

Fig. 6 **a** A triangular mesh of heart with 7,120 elements and **b** the optimized mesh of heart with 3,128 elements

References

1. Caponetto R, Fortuna L, Graziani S, Xibilia MG (1993) Genetic algorithms and applications in system engineering: a survey. Trans Inst Meas Control 15(3):143–156
2. Rudolph G (1994) Convergency analysis of canonical genetic algorithms. IEEE Trans Neural Netw 5(1):96–101
3. Michalewicz Z (1998) Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Berlin Heidelberg New York, ISBN 3-540-60676-9
4. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading
5. Holland JH (1975) Adaptation in natural and artificial system. The University of Michigan Press, Ann Arbor
6. Dorsey RE, Mayer WJ (1995) Genetic algorithms for estimation problems with multiple optima, non-differentiability, and other irregular features. J Bus Econ Stat 13(1):53–66
7. Liu X, Begg DW, Fishwick RJ (1998) Genetic approach to optimal topology/controller design of adaptive structures. Int J Numer Methods Eng 41(5):815–830
8. Absaloms H, Tomikawa T (1996) Surface reconstruction by triangulation using GA. In: Proceedings of the 20th international conference on computers and industrial engineering, Kyongju, Korea, 6–9 October 1996, pp 441–444
9. Qin KH, Wang WP, Gong ML (1997) A genetic algorithm for the minimum weight triangulation. In: Proceedings of the IEEE conference on evolutionary computation, Indianapolis, IN, USA, 13–16 April 1997, pp 541–546
10. Hamann B (1994) A data reduction scheme for triangulated surfaces. Comput Aided Geom Des 11(2):197–214
11. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1993) Mesh optimization. In: ACM SIGGRAPHICS Proceedings, vol 1, pp 19–26
12. Ronfard R, Rossignace J (1996) Full-range approximations of triangulated polyhedra. Proceedings of EUROGRAPHIVS'96. Comput Graph Forum 15(3):67–76
13. Gieng TS, Joy KI, Schussman GL, Trotts IJ (1998) Constructing hierarchies for triangle meshes. IEEE Trans Vis Comput Graph 4(2):145–161