Genetic algorithm transformations for non-orthogonal models

R. Obiała[a], G.M. Seed[a], B.H.V. Topping[a,*], P. Iványi[a], D.E.R. Clark[b]

[a] School of Engineering and Physical Sciences, [b] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, Scotland, UK

Abstract

This paper describes a new method, using genetic algorithms, for transforming non-orthogonal geometric models into orthogonal polygon models. The algorithm is applied to a series of non-orthogonal models and the results compared with those obtained from a fuzzy logic approach [1]. A similarity measure is defined to assess and compare the success of the transformations.

Keywords: Geometric modelling; Genetic algorithms; Transformation; Similarity; Fuzzy logic

1. Introduction

Operations on, and the manipulation of, non-orthogonal models are more complex than the equivalent procedures for orthogonal models. In many branches of engineering the processing of non-orthogonal models is required and considerable effort may be saved by processing an equivalent transformed orthogonal model. This type of transformation enables the solution of complex non-orthogonal problems using simple algorithms applied to the equivalent orthogonal model. For example, partitioning of geometric objects for parallel or distributed processing is usually undertaken on a finite element mesh [2,3]. In some cases, simple algorithms cannot be applied to non-orthogonal problems and hence these transformations are particularly useful. The application of fuzzy logic to orthogonal transformation problems has been shown to be a very promising approach [1]. The genetic algorithm approach described here concentrates on a controlled change in the coordinates of the vertices while at the same time preserving all topological connections between the model entities.

1.1. Similarity

It is important that the transformation should result in orthogonal models that are as similar as possible to the original models. Similarity is a very complex problem that has been recently explored for the purpose of geometry searching engines [4,5]. There are two elements that can be used to define similarity, i.e. edges and vertex angles. The degree of similarity is calculated as an average value as follows:

$$S(P) = \frac{1}{2}\left[\frac{1}{N}\sum_{i=1}^{N} S_{edge}(i) + \frac{1}{N}\sum_{i=1}^{N}\left[1 - \left|\frac{\alpha_{n-ortho}(i) - \alpha_{ortho}(i)}{180}\right|\right]\right] \quad (1)$$

where: $i \in \{1, \ldots, N\}$ refers to a particular edge (or internal angle) of the total number of edges (or internal angles), N, of a model. The first term is a measure of the similarity of the edges. To assess the similarity in a dimensionless form, the ratio by which all the edges are increased (or decreased) in the transformation is calculated as follows:

$$S_{edge}(i) = \begin{cases} \frac{l_{norg}(i)}{l_{orth}(i)} & \text{if } \frac{l_{norg}(i)}{l_{orth}(i)} < 1 \\ \frac{l_{orth}(i)}{l_{norg}(i)} & \text{if } \frac{l_{norg}(i)}{l_{orth}(i)} \geq 1 \end{cases} \quad (2)$$

The second term in Eq. (1) relates to the similarity of the internal angles where $\alpha_{n-ortho}$ is a measure of an angle in the non-orthogonal model, and α_{ortho} is a measure of an angle in the orthogonal model after transformation. The degree of similarity will be equal to 1.0 when the models are identical.

* Corresponding author. E-mail: bhvt@sect.mce.hw.ac.uk

Computational Fluid and Solid Mechanics 2005
K.J. Bathe (Editor)

2. Genetic algorithm implementation

The application of genetic algorithms to the orthogonal transformation of two-dimensional models is described and discussed in this section. This strategy represents a very different approach compared with the fuzzy logic transformation approach presented in reference [1]. The model, specifically a boundary of the model, is treated as a disconnected entity and can be imagined as a number of straight and stiff bars (edges) connected with joints (vertices). Hence the only elements of the model that are mobile are the vertices, which can be moved within a limited range in the two-dimensional coordinate system. Movement of the vertices implies changes in the positions of the edges. The process is illustrated in Fig. 1. The model obtained by moving the vertices is the orthogonal transformed model. Additionally, the new shape of the model should be as similar as possible to the shape of the original model. The coded variables are the coordinates of the vertices of the model. Vertices in a two-dimensional model are described by two coordinates (x, y), which are considered to be independent in all aspects of the calculations. Consequently, the number of variables describing the 2D model is doubled when compared to the number of vertices in the model. Constraints on the positions of the vertices (model control) are formulated to ensure that the vertices are not moved into inappropriate positions by the genetic algorithm. The fitness function used by the genetic algorithms is:

$$F = C + \alpha F_1 + \beta F_2 + \gamma F_3 \tag{3}$$

The first term C is used to ensure that the optimisation is a maximisation problem where the value of C is non-negative. The second term in F controls the orthogonality of the genetic algorithm transformation. The orthogonal character of a model is measured by the values of angles between adjacent edges in the model. Following this reasoning this term in F is defined by the function F_1:

$$F_1 = \sum_{i=1}^{N} f(q_i) \rightarrow max, \tag{4}$$

where N is the number of vertices in the model and $f(\phi_i)$ is a function depending on a particular normalised angle ϕ_i between two adjacent edges. A fourth-order polynomial was used to define this function. The angle ϕ_i is normalised within each sub-domain range $[a_i;\ b_i]$ as follows:

$$q_i = \frac{\phi_i - a_i}{b_i - a_i} \in [0;\ 1] \tag{5}$$

Once the angles are normalised the function $f_n(q_i)$ is defined and used instead of $f(\phi_i)$ in the fitness function. The normalised function is shown below:

$$f_n(q_i) = \begin{cases} 0 & \text{for} \quad \phi_i \in [0;\ 45] \\ (q_i)^4 & \text{for} \quad \phi_i \in [45;\ 90] \\ (1-q_i)^4 & \text{for} \quad \phi_i \in [90;\ 135] \\ (q_i)^4 & \text{for} \quad \phi_i \in [135;\ 180] \\ (1-q_i)^4 & \text{for} \quad \phi_i \in [180;\ 225] \\ (q_i)^4 & \text{for} \quad \phi_i \in [225;\ 270] \\ (1-q_i)^4 & \text{for} \quad \phi_i \in [270;\ 315] \\ 0 & \text{for} \quad \phi_i \in [315;\ 360] \end{cases} \tag{6}$$

Function F_2 in the fitness function is defined as follows:

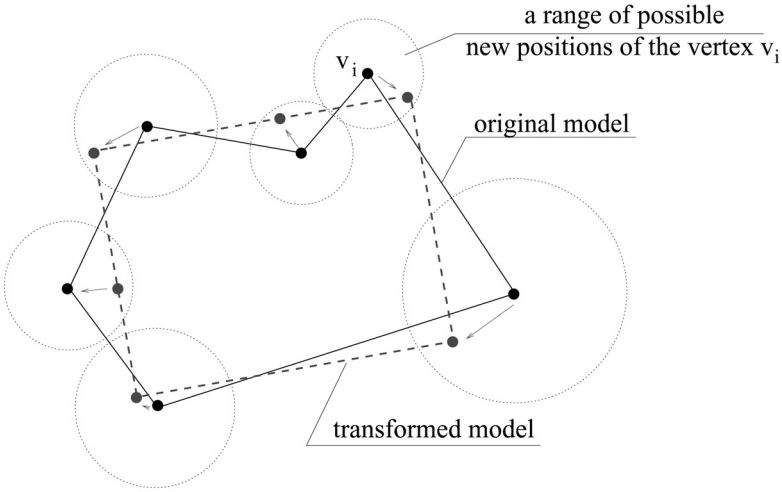


Fig. 1. Model transformation using variation of the coordinates of the vertices.

$$F_2 = \sum_{i=1}^{N} g(\phi_i) \rightarrow min \tag{7}$$

where $g(\phi_i)$ calculates the difference between corresponding angles in the models before and after transformation.

$$g(\phi_i) = \left| \phi_i^{original} - \phi_i^{calculated} \right| \tag{8}$$

To be consistent with the previous definition of the term in F the normalised function is used when defining the function F_2. The value of $g(\phi_i)$ is divided by 180. This is the maximal expected difference between the original and calculated angles. Consequently, the normalised function $g_n(\phi_i)$ provides values within a range [0; 1].

$$g_n(\phi_i) = \frac{g(\phi_i)}{180} \tag{9}$$

The final function F_3 is responsible for assignment of the edges to the coordinate axes and is defined as follows:

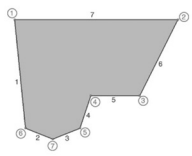
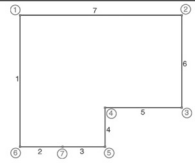
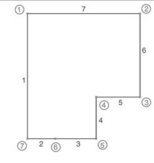
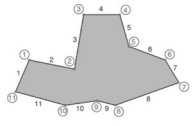
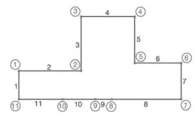
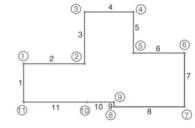
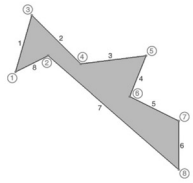
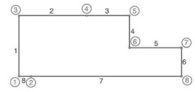
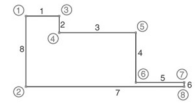
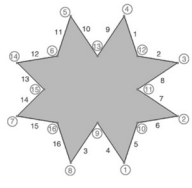
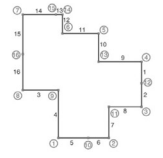
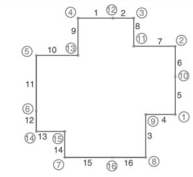
$$F_3 = \sum_{i=0}^{n-1} h(\psi_i) \rightarrow max \tag{10}$$

where ψ_i is the angle between the edge (i) and the x-$axis$. The angles between an edge and both axes linearly depend on each other in the two-dimensional system. That is why it is sufficient to operate with only one of the angles. The angle ψ_i is normalised in the same way as the angle ϕ_i. The angle ψ_i is normalised within the sub-domain range $[a_i; b_i]$ as follows:

$$p_i = \frac{\psi_i - a_i}{b_i - a_i} \in [0;\ 1] \tag{11}$$

As the result of the normalisation the normalised angle p_i is equal to 0 if $\psi_i = a_i$ and is equal to 1 if $\psi_i = b_i$. Consequently, the function h uses the normalised angle p_i defined in each sub-domain (using Eq. 11) instead of the angle ψ_i.

Table 1
Degrees of similarity obtained for the same models using two different transformation methods (fuzzy logic and GA)

Original Model	degree of similarity	Fuzzy Logic Transformed model	degree of similarity	Genetic Algorithm Transformed model
	0.824		0.858	
	0.759		0.778	
	0.796		0.704	
	0.735		0.772	

$$h_n(p_i) = \begin{cases} (1-p_i)^4 & \text{for} \quad \psi_i \in [0;\ 45] \\ (p_i)^4 & \text{for} \quad \psi_i \in [45;\ 90] \\ (1-p_i)^4 & \text{for} \quad \psi_i \in [90;\ 135] \\ (p_i)^4 & \text{for} \quad \psi_i \in [135;\ 180] \\ (1-p_i)^4 & \text{for} \quad \psi_i \in [180;\ 225] \\ (p_i)^4 & \text{for} \quad \psi_i \in [225;\ 270] \\ (1-p_i)^4 & \text{for} \quad \psi_i \in [270;\ 315] \\ (p_i)^4 & \text{for} \quad \psi_i \in [315;\ 360] \end{cases} \tag{12}$$

To ensure that the algorithm performed satisfactorily the values of the weight coefficients α, β and γ in the fitness function (Eq. 3) were found after many trial runs: $C = nb\_edges \cdot |\beta|$, $\alpha = 10\,000$, $\beta = -900$, and $\gamma = 10\,000$, where *nb_edges* is the number of edges in the non-orthogonal model. The C value guarantees that the function F is non-negative.

3. Conclusions: comparison between genetic algorithm and fuzzy logic transformations

Some sample results are shown in Table 1. The table compares the values of the degree of similarity calculated for both the genetic algorithm and fuzzy logic [1] transformation techniques. The degree of similarity is a parameter, which is measured between the original non-orthogonal and the transformed orthogonal models when using the transformation methods. Examination of the results does not show unambiguously that one of the methods always creates orthogonal models with a higher similarity than the other method. On the contrary the degree of similarity is comparable for both methods, which indicates that the models obtained are possibly the best orthogonal models that can be transformed from the given original models.

References

[1] Obiała R, Topping BHV, Seed GM, Clark DER. Transformation of geometric models into orthogonal polyhedra using fuzzy logic. Accepted for publication, Int Journal: Engineering Computations, 2004.

[2] Topping BHV, Khan AI. Parallel Finite Element Computations. Edinburgh, UK: Saxe-Coburg Publications, 1996.

[3] Topping BHV, Khan AI, Sziveri J. Parallel and distributed processing for computational mechanics: an introduction. In: BHV Topping, editor, Parallel and Distributed Processing for Computational Mechanics: Systems and Tools, pp. 1–23. Edinburgh, UK: Saxe-Coburg Publications.

[4] Corney J, Rea H, Clark D, Pritchard J, Breaks M, MacLeod R. Coarse filters for shape matching. IEEE Compurer Graphics and Applications 2002;22(3):65–74.

[5] Funkhouser T, Min P, Kazhdan M, Chen J, Halderman A, Dobkin D, Jacobs D. A search engine for 3D models. ACM Trans Graph ACM Press, 2003;22:83–105.

[6] Obiała R, Topping BHV, Clark DER, Seed GM. Decomposition method applied to orthogonal polyhedra. In: BHV Topping, Z Bittnar, (eds), Proceedings of the Third International Conference on Engineering Computational Technology, Civil-Comp Press, Stirling, UK, Paper 13, 2002.

[7] Obiała R, Iványi P, Topping BHV. Genetic algorithms applied to partitioning for parallel analyses using geometric entities. In: K.J. Bathe, editor, Computational Fluid and Solid Mechanics: 2003, pp. 2078–2081. Massachusetts Insitiute of Technology, Cambridge, USA, Elsevier Science, Amsterdam, 2003.