Fast Flattening Algorithm for Non-developable 3D Surfaces

Yih-Chuan Lin
Department of Computer Science and Information Engineering
National Formosa University, Yunlin, Taiwan 62301

Chia-Hsu Kuo
Department of Computer Science and Information Engineering
National Formosa University, Yunlin, Taiwan 62301

Chun-Ching Wang
Department of Electrical Engineering
National Changhua University of Education, Changhua, Taiwan

Gwo-Long Li
Department of Electrical Engineering
National Dong-Hwa University, Hualien, Taiwan

Abstract *¾ This paper addresses the problem of deriving 2D patterns from triangulated 3D surfaces. Unfolding 3D surfaces into 2D patterns plays an important role in CAD/CAM systems for most industry applications, such as shoe making, apparel manufacturing, or curved ducts (pipes) production, and most of its solutions usually need a large amount of computations. Thus, to speed up the unfolding process would have significant contribution to the field of CAD/CAM research. The paper begins by an extensive literature review on the status of methods, and the main concern would concentrates on how to design a fast algorithm for unfolding 3D surfaces. We paid more attention to improve a method that was proposed by McCartney in 1999. In his work, McCartney gave a more systematic solution description than others do the same thing, and the algorithm performs well in the illustrated cases. However, McCartney's algorithm requires a considerable amount of computation and makes itself a time-consuming process for larger or more complex 3D surfaces. Therefore, in this research we present new concepts about how to use the Max heap tree structure to speed up the surfaces flattening process, in which the frequency of relaxing the small strain energy can be effectively suppressed during the iterative process. This novel flattening methodology would be helpful in terms of both efficiency and precision for most related industries dealing with large-scale and complex 3D surfaces. Simulation results have been conducted for several non-developable surfaces, e.g. elliptical curvature and hyperbolic curvature surfaces, and have shown that the proposed algorithm can offer a significant saving of computations.*

Keywords: 3D Surfaces Flattening, 2D Patterns, 3D CAD/CAM, Heap Tree

1 Introduction

Unfolding 3D surfaces into 2D patterns plays an important role in CAD/CAM systems for most industry applications, such as shoe making, apparel manufacturing, or curved ducts (pipes) production. From the geometric view, the 3D surface can be classified into two categories: 1) Developable surfaces and 2)Non-developable surfaces [1]. For developable surfaces one or both of the principal curvatures must be zero (Fig. 1). Where both principal curvatures are non-zero and lie on the same side of the surface (Fig. 2(a)), this type of curvature is called elliptical. Alternatively, where both principal curvatures are non-zero and lie on opposite sides of the surface, the curvature is called hyperbolic (Fig. 2(b)).

For developable surfaces, Gaussian curvature will be zero everywhere and there is an isometric mapping from the 2D pattern to the 3D surface with no distortion. For non-developable surfaces, the distortion will appear due to the surface can not be isometric mapped into the 2D pattern. In order to reduce the distortion, many algorithms are proposed for reducing this advantage.

In [2], the fundamental theory of this flattening process is based on the Orthogonal Property of Rotation Matrices. The transformation matrices are developed to transform the non-developable surfaces onto the *XY* plane with a boundary constraint and a predetermined flattening direction. Azariadis and Aspragathos [3] consider the derivation of plane developments which correspond with doubly curved surfaces. The first stage of their process is to identify the best generating lines or guide strips on the 3D surface from which to initiate the development process. There is then piecewise development onto the 2D plane of these guide strips. The final stage is the refinement on the overall plane development to satisfy certain criteria.

This paper presents a fast flattening algorithm based on Iterative Strain Energy Relaxation Method (ISERM) [4]. The ISERM algorithm incorporates an energy model in terms of the strain energy required to deform the edges of the triangular mesh. Through the energy relaxation, the deformation of flattened surface can be minimized. An iterative form is used to energy relax for all vertices of the flattened triangle mesh until a defined threshold is reached. This ISERM is computational intensive while the number of triangle mesh is rather huge for flattening. Therefore, this paper presents a fast algorithm to accelerate the flattening speed. Instead of relaxing all the energy of vertices, a

limited number of vertices which have largest energy accumulation should be relaxed. Experimental results show that the proposed algorithm can achieve significant flattening performance improvement with slightly subjective deformation.

In the following, the fundamental principle of Iterative Strain Energy Relaxation Method is overviewed. The proposed Improved Iterative Strain Energy Relaxation (IISERM) algorithm is also described in detail. Some experimental results are shown to demonstrate the performance improvement of proposed algorithm. Finally, a conclusion is made.

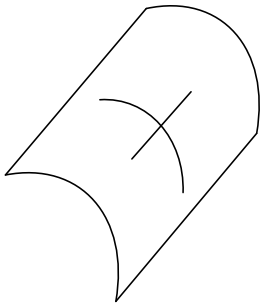

Fig. 1 Developable surface

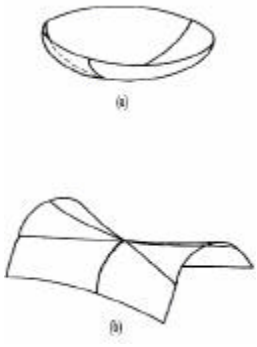

Fig. 2 Non-developable surface (a) elliptical (b) hyperbolic

2 Overview of Iterative Strain Energy Relaxation Method (ISERM) [4-8]

The main idea of this paper is that, unless the surface is developable, during surface flattening some deformation would occur in order to achieve an equivalent 2D pattern. During the flattening process described in the following, edges will be required to strain and so accumulate an intrinsic energy distribution throughout the pattern which will flelect the local departure from the developable property that the surface will embody. Hence if the triangulated surface is a close approximation to the developable surface, the magnitudes of the energy distribution will be very small. Before describing the ISERM algorithm, some notation must to be defined:

1. *Triangle sets*: Triangles from the list T are now considered as belonging to one of three mutually exclusive sets.
2. *Available set* ***V***: There are triangles that have yet to be considered for flattening.
3. *Active set* ***A*** *(ordered)*: These are triangles that share an edge with triangles that have been flattened. As such, these triangles are in a position to be appended to the current set of flattened triangles.
4. *Flattened set* ***F***: These are triangles that have been flattened and have a presence on the 2D flattening.

The flattening procedure of ISERM algorithm is described as follows:

Step 1: Initially all triangles are placed in set **V**. Sets **A** and **F** are empty.

Step 2: Flattening is initiated by detecting the triangular polygon that is located closetst to an averaged position within the surface. This triangle is removed from set ***V*** and placed directly in set ***F***. This seed triangle is arbitrarily laid down on the 2 D plane (x'y') of the flattened pattern-with no strain energy required to do so.

Step3: A search is made of set ***V*** for all triangles that share an edge with the seed polygon. All such triangles are added arbitrarily to the active set ***A***.

Step4: If the active set ***A*** is empty then STOP; Otherwise the next triangle ***T*** is taken from the ordered active set ***A***.

Step5(a): If the third node of triangle ***T*** has not yet been flattened (at least two will already have been flattened as part of the shared edge with a triangle in set ***F***) then unconstrained triangle flattening can proceed.

Step5(b): If the third node of triangle ***T*** has been previously flattened, then constrained triangle flattening and energy relaxation processing will proceed followed by subsequent energy relaxation of the current flattened set of edges.

Step6: Triangle ***T*** is then inserted in set ***F***. A search is made of set ***V*** and if any triangles are found which share an edge with triangle ***T*** then these are added to the end of set ***A***. Go to *Step 4*.

In the ISERM algorithm, the strain energy associated with an edge strut can be described by a function f_{energy} which operates on the positions of the ends of a strut before and after deformation.

$$f_{energy}(P'_1, P'_2, P_1, P_2) = 0.5\frac{(P'_1 P'_2 \quad P_1P_2)^2}{P_1P_2} \tag{1}$$

where P_1 and P_2 indicate the two points which defines an edge and P'_1 and P'_2 indicate the distorted points.

The method adopted here is to take each node (P'_i) in the flattened pattern in turn and attempt to move it an incremental distance Δd in four orthogonal directions P'_{ia}, P'_{ib}, P'_{ic} and P'_{id} as described in Fig. 3. Suppose that node P'_i is connected in the triangulated surface to previously flattened nodes P'_p, P'_q, P'_r and P'_s. Then each node will have an associated energy value of E_i comprising the strain energy of all connected edges. i.e.

$$E_i = f_{energy}(p_i', p_p', p_i, p_p) + f_{energy}(p_i', p_q', p_i, p_q) + f_{energy}(p_i', p_r', p_i, p_r) + f_{energy}(p_i', p_s', p_i, p_s) \tag{2}$$

After each of these test moves, the net change to the overall strain energy of the mesh is evaluated. So for test

position P'_{ia} for example, the energy value E_{ia} is calculated by:

$$E_{ia} = f_{energy}(p'_{ia}, p'_p, p_i, p_p) + f_{energy}(p'_{ia}, p'_q, p_i, p_q) + f_{energy}(p'_{ia}, p'_r, p_i, p_r) + f_{energy}(p'_{ia}, p'_s, p_i, p_s) \quad (3)$$

Similary, E_{ib}, E_{ic} and E_{id} are determined. This enables the maximum energy reduction $?E_i$ to be found where

$$\Delta E_i = \max\{(E_i - E_{ia}), (E_i - E_{ib}), (E_i - E_{ic}), (E_i - E_{id})\} \quad (4)$$

The movement that offers the greatest reduction is then registered with that node. If this maximum energy reduction is not greater than a defined value – called the energy threshold, E_t – then this node will not be considered for movement at this stage. The process involves searching through all nodes in the flattened pattern and finding that node which registers the greast possible reduction in strain energy. The test movement is then implemented for this node and a new value for a possible reduction in strain energy evaluated for this new position and registered with that node. Consequently, all connecting nodes will have to have their maximum energy reduction values re-examined. This process continues until no further reductions greater than E_t is possible.

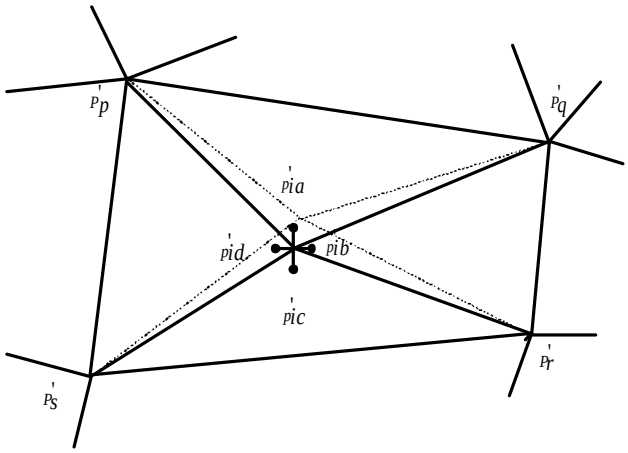

Fig.3 Orthogonal arrangement for trial movements for nodes.

3 Proposed Improved Iterative Strain Energy Relaxation Method (IISERM)

In this section, the proposed fast flattening algorithm is described. From the above section explained, the ISERM algorithm perfroms energy relaxation for the vertices within flatted triangle set while the constrained triangle flattening has been done. This energy relaxing operation is computational intensive if that the items within the flatted triangle set of edges are numerous. By the obsersvation, the energy relaxation may be discarded for some triangles if that the energy relaxation for the triangles are contributed little influence for the results. For this reason, presenting an algorithm to save the unnecessary energy relaxation is the central idea of this paper. The Maximum-Heap tree data structure is used by this paper to accelerate the flattening process. The flattening process of proposed algorithm is based on the ISERM algorithm. But the main different between the ISERM and IISERM is that: for every vertex of flattned triangle, the corresponding energy are calculated and stored in the Maximum-Heap tree. Instead of relaxing energy of all the vertices of the flattened triangles, some important vertices which maintaining maximum energy are selected to relax.

Fig. 4 shows a proportion of a 3D surface with 16 vertices and 18 triangle meshes. After the flattening, the energy for all vertices can be structured into a Maximum-Heap tree (Fig. 5). In the Fig.4, assume that the energy of vertex 1 (E1) is maximum. The energy relaxation process selects some vertices which has largest energy accumulation from the Maximum-Heap tree for flattening. The proposed algorithm is described as follows:

Step1: Select a triangle mesh for flattening from the un-flattened triangle mesh set randomly. Flatten that triangle mesh and calculates the energy for each vertex. Add the energy of each vertex into the Maximum-Heap tree.

Step2: Find the other triangle meshes that share the same edge with the flattened triangle mesh and add into a queue (Q) for waiting to flatten.

Step3: If the Q is empty then STOP; Otherwise, select a triangle mesh from the Q and goto Step4.

Step4: If that the triangle mesh is a constrained triangle then goto Step5; Otherwise, goto Step6.

Step5: Flatten the triangle mesh and calculates the energy. Add the energy into the Maximum-Heap tree and goto Step7.

Step6: Calculate the distance ($?d$) between the location of the new decided point and the location of the original point. If $?d$ less than a *threshold* then goto Step7; Otherwise, Flatten the triangle mesh and calculates the energy. Add the energy into the Maximum-Heap tree and goto Step2.

Step7: Perform the energy relaxation for some vertices that maintaining maximum energy in the Maximum-Heap tree. Goto Step8.

Step8: After the energy relaxation, recalculate the energy for the energy relaxed vertices and modify the Maximum-Heap tree. Goto Step2.

As above mentioned, the proposed IISERM selects some vertices which have largest energy accumulation to relax. How many vertices should be selected to relax is another issue of this paper. In this paper, two methods are proposed and the detail is described as follows.

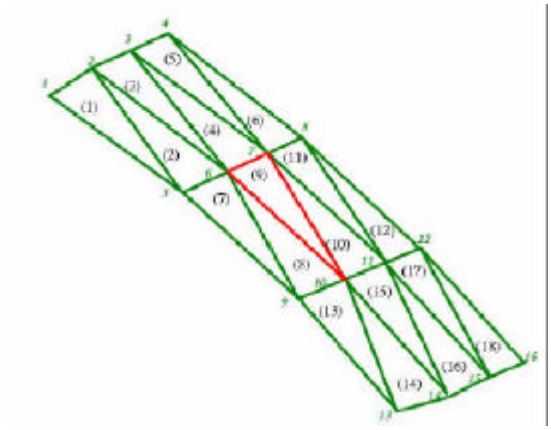

Fig. 4 A proportion of 3D surface

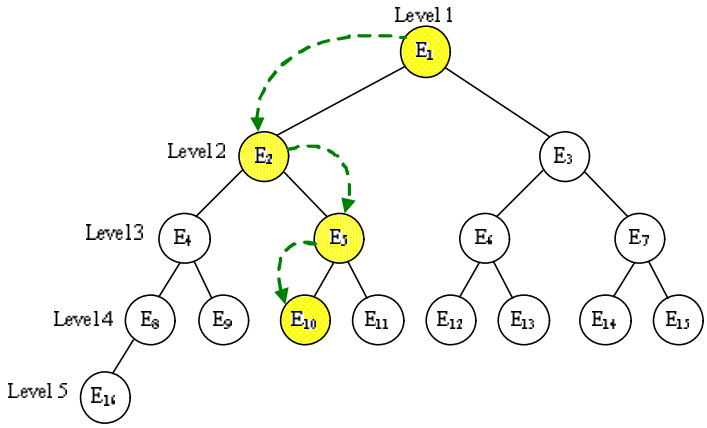

Fig. 5 The Maximum-Heap tree representation for the energy

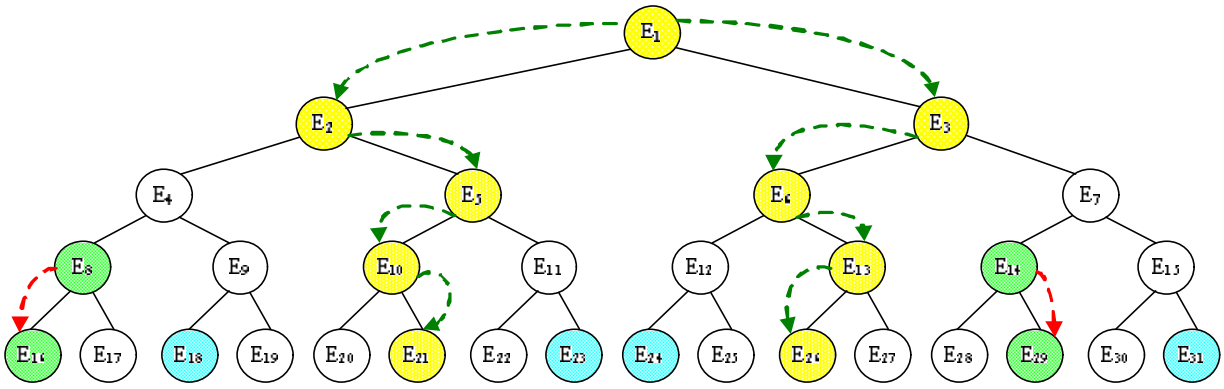

Fig. 6 Illustration of Multi-Path energy relaxation

3.1 Single path energy relaxation

As show in Fig.5, only one vertex is selected for energy relaxing at each level. For the first level, the energy relaxation is essential because the first level only one vertex and such vertex have larger energy accumulation than the other vertices. In the second level, one needs to determine which energy of vertex is largest and to perform the energy relaxation for the selected vertex. Afterward only two vertices need to be checked and only one vertex need for flattening at each level.

3.2 Multi-path energy relaxation

Fig. 6 shows the multi-path energy relaxation algorithm. For the level 1-3, the vertices on the two paths need to be relaxed. Afterward the paths are increased with the level and the number of paths is calculated by the equation 5. Therefore, all the vertices on the paths need to be relaxed.

$$path = 2^{level-2} \Big|, level > 3 \qquad (5)$$

4 Results

This section presents the experimental results of proposed IISERM algorithm with comparisons to ISERM. The objective and subjective quality comparison are also shown in this section. Fig.7 shows the comparison of the flattening time. The horizontal axis is the number of vertices and the vertical axis indicate the total flattening time. It is obvious that the proposed IISERM algorithm results in significant flattening time improvement. Since the ISERM is an iterative procedure for the energy relaxing, the flattening time consumption increases significantly with the growth of the number of vertices. The proposed IISERM algorithm only selects some vertices for energy relaxing. Therefore, the proposed IISERM algorithm can efficiently save the flattening time. Fig. 8 shows the energy accumulation after the flattening. Since most of the vertices which maintaining largest energy have been energy relaxed by the proposed IISERM algorithm. Therefore, the difference of remained energy between proposed IISERM and ISERM are very small. The subjective quality is shown in Figs.9-12. Fig.9 shows an original 3D surface and the flattened result is shown in Fig.10. Fig.11 shows an original 3D surface for another complex case. The flattened result is also shown in Fig.12.

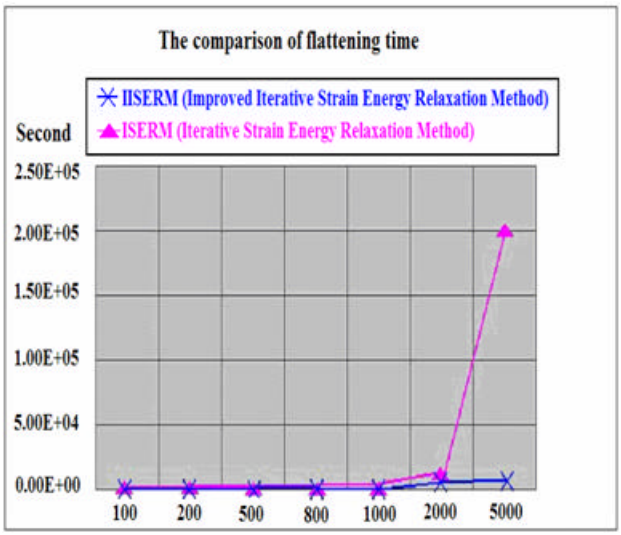

Fig. 7 The comparison of flattening time

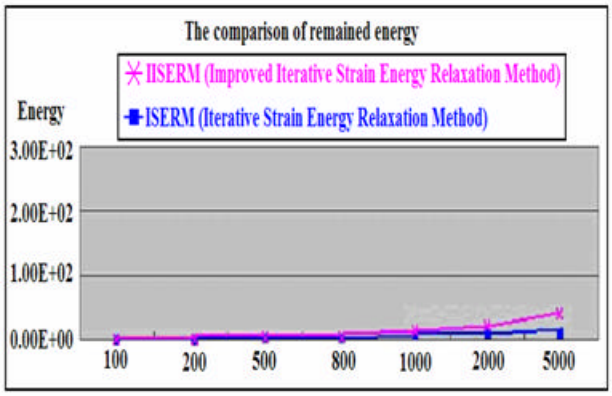

Fig. 8 The comparison of remained energy

Fig. 8 Original 3D surface

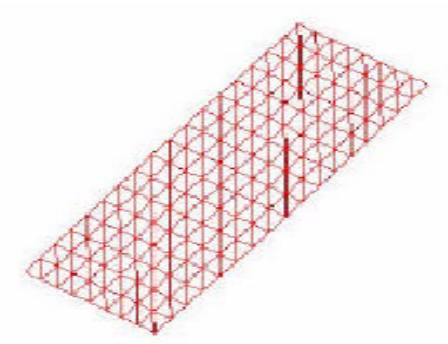

Fig.9 The result after the flattening

Fig. 10 Original 3D surface

Fig. 11 The result after the flattening

5 Conclusion

This paper presents a fast algorithm for 3D flattening. Instead of relaxing the energy of all vertices of the flattened triangle, the proposed algorithm using the Maximum-Heap tree for storing the energy of flattened vertices and select some vertices which have largest energy accumulation to flatten. Experimental result shows that the proposed IISERM can improve the coding speed significantly with slightly deformation while compared to the ISERM algorithm. This performance improvement is useful for the production in industry.

ACKNOWLEDGEMENT

The authors would like to acknowledge the funding support in part from the National Science Council, Taiwan under the contract NSC-94-2213-E-150-030, and the National Formosa University under the contract TCS93217.

References

[1] E. D. Bloch, *A First Course in Geometric Topology and Differential Geometry*, Birkhäuser Boston 1997.

[2] M. D. Shieh, C. C. Yang, "*Computer Aided System for 2D Flatting of 3D Apparel – Non-developable Surfaces*," Master Thesis, National Chen-Kong University, 2002

[3] Azariadis P, Asparagathos N. "Design of plane developments of doubly curved surfaces," *Computer-Aided Design* 1997; 29(10):675-685

[4] J. McCartney, B. K. Hinds, and B. L. Seow, "The Flattening of Triangulated Surfaces Incorporating Darts and Gussets," *Computer-Aided Design*, Vol. 31, pp. 249-260, 1999.

[5] B. K. Hinds, J. McCartney, and G. Woods, "Pattern Development for 3D Surfaces," *Computer-Aided Design*, Vol. 23, No. 8, pp. 583-592, 1991.

[6] B. K. Hinds and J. McCartney, "Interactive Garment Design," *The Visual Computer*, pp. 53-61, 1990.

[7] B. K. Hinds, J. McCartney, C. Hadden, and J. Diamond, "3D CAD for Garment Design," *International Journal of Clothing Science and Technology*, Vol. 4, No. 4, pp. 6-14, 1992.

[8] J. N. Reddy, *An Introduction to Finite Element Method, 2d edition*, McGraw-Hill 1993.