International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:5, No:7, 2011

# Surface Flattening based on Linear-Elastic Finite Element Method

Wen-liang Chen, Peng Wei and Yidong Bao

***Abstract*—**This paper presents a linear-elastic finite element method based flattening algorithm for three dimensional triangular surfaces. First, an intrinsic characteristic preserving method is used to obtain the initial developing graph, which preserves the angles and length ratios between two adjacent edges. Then, an iterative equation is established based on linear-elastic finite element method and the flattening result with an equilibrium state of internal force is obtained by solving this iterative equation. The results show that complex surfaces can be dealt with this proposed method, which is an efficient tool for the applications in computer aided design, such as mould design.

***Keywords*—**Triangular mesh; Surface flattening; Finite element method; Linear-elastic deformation

## I. Introduction

3D surface flattening is currently widely used in the field of CAD and computational geometry. Flat patterns generated from original 3D surfaces can be used for reverse engineering to approximate the 3D objects, and are also critical for creating a texture mapping. In practice, 3D surfaces can be classified developable surface and non-developable surface according to their malleability, and 3D surfaces are usually non-developable. The development of a non-developable surface is a complicated process and distortion is inevitable during surface flattening.Up to now, many researchers developed a lot of methods to solve the flattening problem. The most common methods used in surface flattening are energy based method, energy functions are created by physical model, and flattening results can be obtained by solving a minimum energy or equilibrium state. McCartney et al. [1] presented a flattening algorithm by deforming edges of the triangular mesh with an energy model in terms of the strain energy. In addition, the process of flattening was capable of handling the insertion of darts and gussets. Based on the same methodology, Wang et al. [2] presented a method for three-dimensional surface flattening, a mass spring model based energy function was created and penalty function is applied to recover the overlapped area. Recently, Wang et al. [3] proposed another method based on fitting a woven-like mesh model on a 3D freeform surface. It consists of two steps, the first step is to insert the seeds in the discrete geodesic curve generation algorithm for 3D surface fitting, and then to establish the planar coordinate mapping between the 3D surface and its counterpart in the plane by geodesic interpolation of the mappings. Strain energy minimization was emphasized in both seed insertion and the mapping procedure. Zhong et al. [4] also presented a mass spring model based method to divide the 3D surface into nearly developable charts, and then flatten these charts.Besides these energy based methods, Parida et al [5] proposed an algorithm to develop complex surfaces. Their algorithm first obtains an approximate planar graph, and then reorients cracks and overlapping parts in the developed plane to satisfy orientation constraints. Their algorithm might generate many accumulative errors and cracks. Floater [6] presented a parameterization method based on graph theory. In this method, the position of each vertex in the flattened pattern was determined by solving a linear system based on convex combinations. A major disadvantage of this method is that it requires the boundary of the two-dimensional mesh to be predefined and convex. Sheffer et al. [7] presented an angle based method to compute planar triangulations for surface parameterization. This method focus on directly optimizing the geometric distortion metric of angles, obtains the flattening result by solving a nonlinear system. It is usually require extensive computation due to highly non-linear. L´evy et al. [8] presented a quasi-conformal parameterization method based on a least-square approximation. This method used an objective function to minimize angle deformation, and a complex surface was decomposed into charts with natural shapes. However, the chart was smaller when the boundary of the surface self intersected in the texture space. Liu et al. [9] proposed a local/global algorithm, which combines a local mapping of each triangle to the plane, with a global "stitch" operation of all triangles, involving a sparse linear system. Our algorithm can be viewed as a variant of recent energy based surface flattening methods. All these methods generally consist of two steps. The first step is to calculate an initial developing pattern, and the second step is to reduce the difference between initial developing pattern and the 3D surface by an iterative method. Therefore, the main issues we concerned about are the appropriate initial developing pattern and the energy based iterative method.

Wenliang Chen, male, Ph.D, professor, College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics,Nanjing,210016,China(phone:+8625-84892500;fax:+8625-84891501; e-mail: cwlme@nuaa.edu.cn).

## II. Problem Statement

The input of our algorithm is 3D surfaces described by triangular meshes. Triangular meshes M = (V, E, T) can be represented as a set of three dimensional vertices, edges and faces. $V = \{v_i\}(i=0,1,\ldots,n\text{-}1) \in R^3$ is the set of vertices, $n$ is the number of vertices; edges $E = \{e_i\}$, $e = \{v_i, v_j\}$ is composed of two adjacent vertices $v_i$ and $v_j$; triangular faces $T = \{t_i\}$, $t = \{e_i, e_j, e_k\}$ is composed of three adjacent edges. The adjacent vertices of vertex $v_i$ are denoted by $N(v_i) = \{v_j | \ (v_i, v_j) \in E\}$, the number of adjacent vertices is called the degree of vertex $v_i$, and is denoted as $|N(v_i)|$. In order to flatten the triangular mesh M, the process of our proposed algorithm for 3D surface flattening is described in detail as follow.

Step 1: Import three dimensional triangular mesh *M*, and some preparations for surface flattening are required, such as hole filling [10].

Step 2: Obtain the initial planar graph by keeping the intrinsic characteristics of original triangular mesh *M*.

Step 3: Create the linear-elastic finite element energy model, build the planar stiffness matrix, and calculate the internal force for every node based on its displacement.

Step 4: Obtain the displacement of every node in initial planar graph by solve the internal force equilibrium function.

Step 5: Calculate the value of internal force and displacement of every node, if they meet the convergence criterion, then output the flattening results, otherwise go back to step 3.

## III. Surface Flattening using a Elastic Force Equilibrium Model

In this section, a linear method based on keeping the intrinsic characteristics of angles and length ratios is introduced to create the initial planar graph, and an iterative energy release method based on linear-elastic finite element method is used to obtain the final flattening result.

### *1) Initial planar graph created by preserving intrinsic characteristic*

It is important to create an appropriate initial planar graph, means that the initial planar graph is close to the global optimization and is little time-consuming. In order to get an appropriate initial planar graph, a linear method is adopted by preserving the intrinsic characteristics of angles and length ratios between two adjacent edges [11]. For a triangle $T = \triangle p_1p_2p_3$ in triangular mesh *M*, the adjacent edge of $p_1p_2$ is $p_1p_3$ in counterclockwise; $\alpha_{213}$ is the orientation angle between $p_1p_2$ and $p_1p_3$; $\lambda_{213}$ is the length ratio between $p_1p_2$ and $p_1p_3$, $\lambda_{213} = \frac{\| p_1p_3 \|}{\| p_1p_2 \|}$; $p_1p_3$ is considered to be a edge which firstly rotate $\alpha_{213}$ from $p_1p_2$ and then scale $\lambda_{213}$, the vector equation can be obtained:

$$\overrightarrow{p_1p_3} = \lambda \boldsymbol{R}(\alpha) \overrightarrow{p_1p_2} \quad | \tag{1}$$

where $\boldsymbol{R}(\alpha)$ is a rotation matrix:

$$\boldsymbol{R}(\alpha) = \begin{bmatrix} \cos\alpha, & -\sin\alpha \\ \sin\alpha, & \cos\alpha \end{bmatrix} \tag{2}$$

The equation (1) is invariant to geometric transformation such as translation, rotation and scaling, it determines the relative positions with respect to its first order neighborhood vertices. In two dimensional space, $v_i = (x_i, y_i)$, $i = 1,2,3$, equation (1) can be written as:

$$\begin{bmatrix} \lambda_{213}\cos\alpha_{213} - 1 & \lambda_{213}\sin\alpha_{213} & \lambda_{213}\cos\alpha_{213} & \lambda_{213}\sin\alpha_{213} & 1 & 0 \\ \lambda_{213}\sin\alpha_{213} & \lambda_{213}\cos\alpha_{213} - 1 & \lambda_{213}\sin\alpha_{213} & \lambda_{213}\cos\alpha_{213} & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3}$$

Equation (3) can be obtained for arbitrary two adjacent edges in triangular mesh *M*, then the simultaneous equations can be obtained equivalent to the matrix form:

$$\boldsymbol{AX}=\boldsymbol{0} \tag{4}$$

Equation (4) only preserves the relative position in triangular mesh *M*, hence, global positions for some constraint nodes are required to specify. We generally specify the planar coordinates of a flat mesh or the planar coordinates of boundary nodes. Let $P_s = (x_s, y_s)$ $(s=0,1,\ldots,l\text{-}1) \in R^2$ be a set of constraint nodes, $l$ is the number of constraint nodes, we can get the position constraint equations: $x_s = x_t$, $y_s = y_t$, $(s,t=0,1,\ldots,l\text{-}1)$, which is equivalent to the matrix form:

$$\boldsymbol{C}_A\boldsymbol{X} - \boldsymbol{R} = \boldsymbol{0} \tag{5}$$

where matrix $\boldsymbol{C}_A$ is a sparse matrix with $2l$ row and $2n$ column; $\boldsymbol{R}$ is $2l \times 1$ matrix composed of the coordinates of vertices of $P_t$.

Our objective is to create the planar graph by preserving the angles and length ratios between two adjacent edges, and at the same time let the planar graph satisfy those position constrains. Penalty method is used here to achieve our objective, therefore, the penalty function is

$$E(\boldsymbol{X}) = (\boldsymbol{AX})^T(\boldsymbol{AX}) + \theta(\boldsymbol{C}_A\boldsymbol{X} - \boldsymbol{R})^T(\boldsymbol{C}_A\boldsymbol{X} - \boldsymbol{R}) \tag{6}$$

where $\theta$ is the penalty number, Put $\boldsymbol{A}_K = \boldsymbol{A}^T\boldsymbol{A} + \theta\boldsymbol{C}_A^T\boldsymbol{C}_A$ and the above penalty function can be reformulated as the following:

$$E(\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{A}_K\boldsymbol{X} - 2\theta\boldsymbol{X}^T\boldsymbol{C}_A^T\boldsymbol{R} + \theta\boldsymbol{R}^T\boldsymbol{R} \tag{7}$$

Let $\frac{\partial E(\boldsymbol{X})}{\partial \boldsymbol{X}} = 0$, it follows that

$$\boldsymbol{A}_K\boldsymbol{X} = \theta\boldsymbol{C}_A^T\boldsymbol{R} \tag{8}$$

The matrix $\boldsymbol{A}$, $\boldsymbol{K}$, and $\boldsymbol{C}_A$ are sparse matrix, therefore the coefficient matrix $\boldsymbol{A}_K$ of function (8) is a sparse matrix, a fast solution for sparse equation (8) cab be derived by TAUCS lib [12].

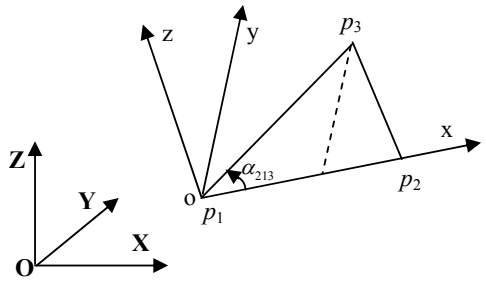


Fig.1 Spatial triangle and its corresponding local coordinate system

*2) Flattening optimization based on linear-elastic finite element method*

For the triangle T=△ $p_1p_2p_3$ given in section 3.1, we can establish local coordinate system *oxyz*. Let the origin of *oxyz* placed in point $p_1$, axis x orientates along $p_1p_2$, axis y is perpendicular to axis x and points to the side of point $p_3$, the local coordinates of triangle T can be denoted as $P_i=(x_i y_i)$, $i$ = 1,2,3. The coordinate transformation matrix between local coordinate system and global coordinate system can be obtained directly

$$\boldsymbol{T}=\begin{bmatrix}\boldsymbol{\lambda} & 0 & 0\\ 0 & \boldsymbol{\lambda} & 0\\ 0 & 0 & \boldsymbol{\lambda}\end{bmatrix} \tag{9}$$

where $\boldsymbol{\lambda}=\begin{bmatrix}l_{ox} & m_{ox} & n_{ox}\\ l_{oy} & m_{oy} & n_{oy}\\ l_{oz} & m_{oz} & n_{oz}\end{bmatrix}$, $l_{ox}$ , $m_{ox}$ , $n_{ox}$ is the cosine value of axis ox corresponding to axis X , Y and Z respectively in global coordinate system, $l_{oy}$ , $m_{oy}$ , $n_{oy}$ , $l_{oz}$ , $m_{oz}$ , $n_{oz}$ can be calculated similarly.

The local coordinate system $o_0x_0y_0z_0$ is established in 3D triangular mesh, while $o_nx_ny_nz_n$ is the local coordinate system in planar mesh corresponding to the triangle in 3D. The local coordinates of vertices in $o_0x_0y_0z_0$ and $o_nx_ny_nz_n$ are x(3)、y(3) and $x_n(3)$、$y_n(3)$ respectively. The stiffness matrix of triangular element e in local coordinate system is expressed as:

$$\boldsymbol{k}^{e}=\int_{v^e}\boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}dv=\boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}hA \tag{10}$$

where matrix $\boldsymbol{B}$ is the strain matrix of triangular element; matrix $\boldsymbol{D}$ is the elastic matrix defined in [13]; $h$ is the thickness of the element; $A$ is the area of triangular element. The strain matrix $\boldsymbol{B}$ can be written as:

$$\boldsymbol{B}=\frac{1}{2A}\begin{bmatrix}y_{23} & 0 & 0 & y_{31} & 0 & 0 & y_{12} & 0 & 0\\ 0 & x_{32} & 0 & 0 & x_{13} & 0 & 0 & x_{21} & 0\\ x_{32} & y_{23} & 0 & x_{13} & y_{31} & 0 & x_{21} & y_{12} & 0\end{bmatrix} \tag{11}$$

where $y_{ij}=y_i-y_j, x_{ij}=x_i-x_j, (i,j=1,2,3)$。The stiffness matrix of element e in global coordinate system:

$$\boldsymbol{K}^{e}=\boldsymbol{T}^{T}\boldsymbol{k}^{e}\boldsymbol{T} \tag{12}$$

where matrix $\boldsymbol{T}$ is the coordinate transformation matrix defined in equation (9).

Suppose that the displacement vector triangular element in local coordinate system is $\boldsymbol{q}$,

$$\boldsymbol{q}=[\Delta x_1,\ y_1,\ 0,\ x_2,\ y_2,\ 0,\ x_3,\ y_3,\ 0]^{T}$$

Let the local coordinate system $o_nx_ny_nz_n$ is placed at $o_0x_0y_0z_0$, therefore, the value of displacement vector $\boldsymbol{q}$ can be calculated as follows

$$\begin{cases}\Delta x_i=x_n(i) & x(i)\\ \Delta y_i=y_n(i) & y(i)\end{cases}\quad i=1,2,3 \tag{13}$$

Therefore, the stress of triangular element can be written as:

$$\sigma=\begin{Bmatrix}\sigma_u\\ \sigma_v\\ \tau_{uv}\end{Bmatrix}=\boldsymbol{DBq} \tag{14}$$

Finally, the internal force of triangular element e in global coordinate system can be calculated:

$$\boldsymbol{F}_{in}=\boldsymbol{T}^T\int_{v^e}\boldsymbol{B}^T\sigma dv=\boldsymbol{T}^T\boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}\{\boldsymbol{q}\}hA \tag{15}$$

The states of three dimensional triangular mesh and planar mesh are considered, the linear-elastic deformation from three dimensional triangular mesh to planar mesh will cause node residual internal forces, and which has ignored the impact of external loads [14]. In order to obtain an equilibrium state of the internal force, Newton-Raphson (abbreviated as N_R) iterative method is used, and we can obtain the N_R iterative form:

$$\begin{cases}\boldsymbol{K}^{i}\Delta\boldsymbol{q}^{i}=\boldsymbol{F}_{\text{in}}(\boldsymbol{q}^{i})\\ \boldsymbol{q}^{i+1}=\boldsymbol{q}^{i}+\omega\Delta\boldsymbol{q}^{i}\end{cases} \tag{16}$$

where matrix $\boldsymbol{K}$ is the global stiffness matrix of triangular mesh $M$ in global coordinate system, $\boldsymbol{F}_{in}(q)$ can be calculated by equation(15) for every element; where $0<\omega<1$ denotes the relaxation factor; $\Delta q^i$ is the displacement of the $i$th iterative step.

The purpose of iterative solution is to achieve a relative equilibrium in final state. Here, we adopt the displacement convergence criterion and internal force convergence criterion to determine the convergence and the convergence criterion are determined as:

$$\left\|\Delta q^{i}\right\|=\frac{1}{n}\sqrt{\sum_{k=1}^{n}(\Delta q_k^{i})^2}\quad \varepsilon \quad < \tag{17}$$

$$\left\|F_{\text{in}}(q^{i})\right\|=\frac{1}{n}\sqrt{\sum_{k=1}^{n}\left(F_{\text{in}\,k}(q^{i})\right)^2}<\gamma \tag{18}$$

where $n$ is the number of nodes, $\varepsilon$ and $\gamma$ is the given precisions, the precision of displacement convergence criterion is $10^{-2}$, while $\varepsilon$ is $10^{-3}$ for internal force convergence criterion. The iteration will be terminating if one of the convergence criterions is satisfied.

## IV. Experimental Results

In this paper, the described mesh projection algorithm was realized using object-oriented programming and C++ language, and was implemented on a 3.00GHz Pentium(R) 4 computer with 2.00GB memory. The present method is applied in surface flattening of different engineering fields such as model design,

mesh regeneration. In the following, we show several flattening examples using our method, and the data statistics and timings for these examples will be given.

The first example in Fig.2 shows a three ball surface flattened by the method. Fig.2(a) shows the three dimensional model with 1925 triangle facets and 790 nodes. Fig.2(b) shows the initial planar graph by preserving the intrinsic characteristics of angles and length ratios between two adjacent edges. After the internal force is released by linear-elastic finite element method, the resulting flattening surface with low distortion is shown in Fig.2(c).

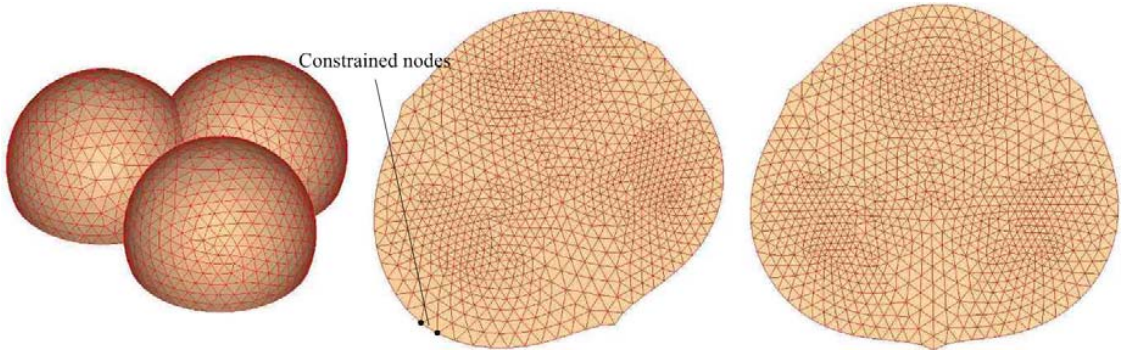


Fig. 2 Flattening of three ball surface (a) three ball model, (b) initial planar graph by intrinsic characteristic preserving, (c) flattening results by linear-elastic finite element method

We tested the sensitivity of our algorithm to different types of constraints defined in initial planar graph generation. Fig.3(a) shows the side floor model with 12174 triangle facets and 6262 nodes. Fig.3(b) shows the initial planar graph with constrained region, and the final flattening surface after 8 times iterations is shown in Fig.3(c). Fig.3(d) shows the initial planar graph with several constrained nodes, and the final resulting flattening surface after 10 times iterations is shown in Fig.3(e). From Fig.3, we can see that the constraints only affect the times of iteration in the progress of internal force release iterative, and the final resulting flattening surface is not sensitive to the constraints.

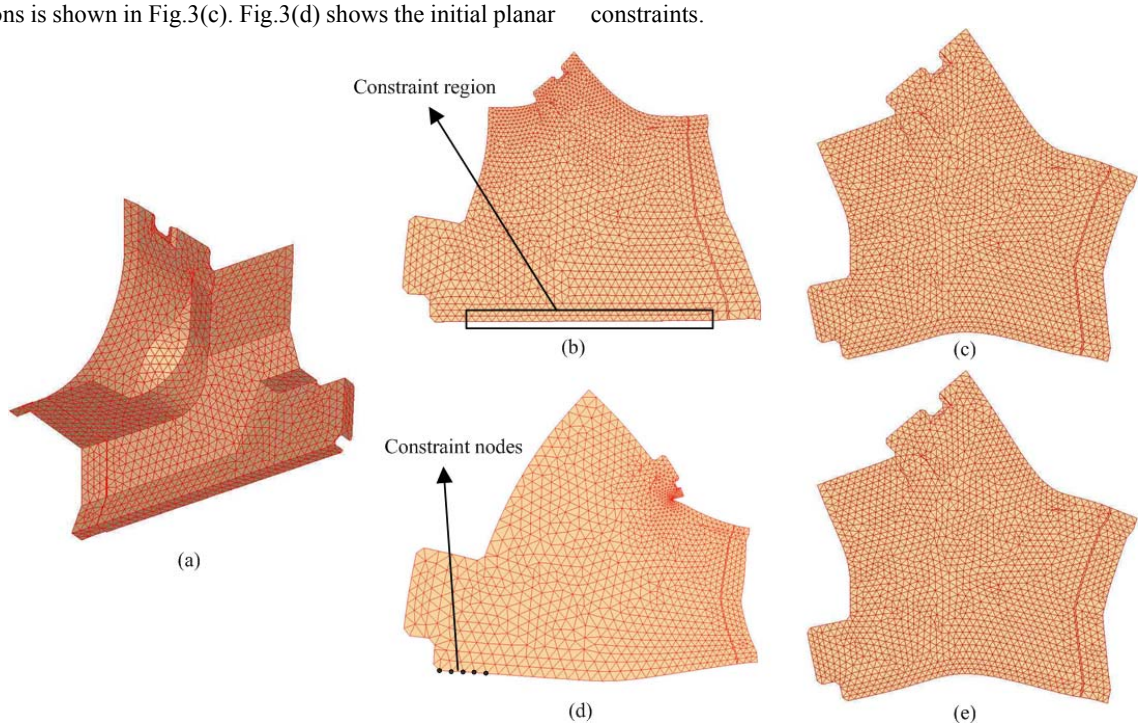


Fig.3. Flattening of side floor with different constraints (a) side floor model, (b) initial planar graph by constraint region, (c) flattening result with result of Fig.3(b), (d) initial planar graph by constraint nodes, (c) flattening result with result of Fig.3(d)

Fig.4 shows the flattening result of a fender surface using in mould design of auto body. Fig.4(a) shows the three dimensional model with 28954 triangle facets and 15096 nodes, the model is very complex and many elements are undercut. The flattening result with 10 times iteration is shown in Fig.4(b). From Fig.4, we can see
that flattening result is similar to the original model with low distortion. Table.1 shows the data statistics and runtime for several model presented in this paper. The initial time refers to the initial planar graph generation by our intrinsic characteristics preserving method, while the iterative time refers to the internal force release by Newton-Raphson iteration. The runtime of side floor refers to the initial planar graph with constrained region shown in Fig(b). In order to confirm the effectiveness of our algorithm, we adopt angular distortion and area distortion to evaluate the flattening distortion. Similar to [6], the angular distortion is $\frac{E(\alpha)}{3n_f}$, where $E(\alpha)$ is defined in [6] as follows.

$$E(\alpha)=\sum_{j=0}^{n_f}\sum_{k\in T_j}\frac{1}{\omega_k^j}(\alpha_k^j-\beta_k^j)^2 \qquad (20)$$

(a) (b)

Fig.4 Flattening of fender (a) three dimensional surface, (b) flattening result

Where $n_f$ is the facet number; $\alpha_k^j$ is the planar angles and $\beta_k^j$ is the original angles; the weight $\omega_k^j$ are set to $1/\beta_k^{j2}$. Area distortions in this paper are measured by method proposed in [6].

TABLE.I
STATISTICS AND RUNTIME

| | Mesh data | | Runtime | | Distortion | |
|---|---|---|---|---|---|---|
| Model | Node | Facet | Initial | Iterative | Angle | Area |
| Three balls | 790 | 1925 | 0.03 | 0.21 | 0.00289 | 0.0247 |
| Side floor | 6262 | 12174 | 0.21 | 1.95 | 0.00575 | 0.0156 |
| Fender | 15096 | 28954 | 1.03 | 8.92 | 0.00781 | 0.0129 |

## V. CONCLUSIONS AND FUTURE WORK

An efficient flattening algorithm for triangulated surface is presented in this paper. An intrinsic characteristics preserving method is introduced in order to generate the initial developing graph. The method is little time consuming and the generated initial developing graph is close to the global optimization. Then, the linear-elastic finite element method is introduced to release the internal force. Unified and explicit formulas are derived to compute the flattening results, and the presented method is easy to comprehend and to implement. Numerical results show that the flattening results obtained by our method have small distortion compare with the three dimensional meshes, and can deal with most complex surfaces efficiently and robustly.In future work, we will consider the surface flattening problem with conformal mapping technology. The conformal mapping technology will reduce the distortion during flattening process, and a new equilibrium state is taken into account to speed up the convergence..

## REFERENCES

[1] McCartney J, Hinds BK, Seow BL. The flattening of triangulated surfaces incorporating darts and gussets. Computer Aided Design 1999, 31(4):249–260.
[2] Wang CCL, Kai T, Benjamin ML. Freeform surface flattening based on fitting a woven mesh model. Computer Aided Design 2005, 37(8): 799–814.
[3] Wang CCL, Chen SSF, Yuen MMF. Surface flattening based on energy model. Computer Aided Design 2002, 34(11): 823–833.
[4] Yueqi Zhong, Bugao Xu. A physically based method for triangulated surface flattening. Computer Aided Design, 2006, 38(10): 1062-1073.
[5] Parida L, Mudur SP. Constraint-satisfying planar development of complex surfaces. Comput Aided Des 1993, 25(4): 225–32.
[6] Floater MS. Parameterization and smooth approximation of surface triangulations. Computer Aided Geometry Design 1997, 14(3): 231–250.
[7] Sheffer A, Lévy B, Mogilnitsky M,*et al*. ABF++: fast and robust angle based flattening. ACM Trans Graph 2005, 24(2):311-330.
[8] L´evy B, Petitjean S, Ray N, et al. Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics, 2002, 21(3): 362-371.
[9] Liu LG, Zhang L, Xu Y, et al. A local/global approach to mesh parameterization [J]. Computer Graphics Forum, 2008, 27(5): 495-504.
[10] Chen WL, Zhang S, Jin XB. Research on the algorithm of hole repairing of finite element mesh. Chinese journal of computers, 2005, 28(6): 1068-1071.
[11] Sederberg T W, Gao P, Wang G J, *et al*. 2-D shape blending: an intrinsic

solution to the vertex path problem. Proceedings of SIGGRAPH. Los Angeles: ACM, 1993: 15-18.
[12] Toledo S. Taucs: a library of sparse linear solvers [EB/PL]. (2003-9-4) [2010-3-10]. http://www.tau.ac.il- /~stoledo/taucs/.
[13] Wang XC. Finite element method [M]. Beijing: Tsinghua University Press, 2003.
[14] Bao YD. Research on one step inverse forming fem and crash simulation of auto body part. Ph.D. thesis. Changchun: Jilin University; 2005.