An improved boundary extraction method of STL model based on edge curvature estimation

Jingbin Hao[1]*, Liang Fang[2], Haifeng Yang[1]

[1]*College of Mechanical and Electrical Engineering, China University of Mining and Technology, Xuzhou, 221116, China*

[2]*State Key Laboratory of Mechanical Behaviour of Metals, Xi'an Jiaotong University, Xi'an, 710049, China*

Received 1 August 2014, www.cmnt.lv

Abstract

To efficiently extract feature boundaries of the STL model, an improved method is proposed based on edge curvature estimation. Three curvature parameters (dihedral angle, perimeter ration and convexity) are used to estimate the surface curvature information of the STL model. Genetic Algorithm (GA) is used to determinate the threshold of feature edges. The extracted feature edges are grouped and filtered using the best-fit plane (BFP), which is calculated by Least Square Method (LSM). The Dijkstra's algorithm is used to close the incomplete feature boundaries. Several experimental results demonstrate that the amount of feature edges is significantly reduced, and useful feature edges are reserved to construct feature boundaries. The improved boundary extraction method has important significance in decomposing large complex STL models meaningfully.

Keywords: boundary extraction, curvature estimation, STL model, genetic algorithm, least square method

1 Introduction

One common strategy for dealing with a large complex model is to decompose it into smaller and simpler sub-models. As STL files use triangle meshes to represent the surface of a solid model, the core issue is how to decompose and reconstruct triangle meshes. To optimally partition the STL model, we first need to extract boundaries between the meaningful sub-models [1, 2].
There are three main approaches that can be used for the feature boundary extraction of an STL model.

1) Vertex-based method: The definition of a sub-model is the one in which regions consist of connected vertices which have the same curvature value (within a tolerance) [3, 4].

2) Edge-based method: A feature edge is an edge shared by two faces whose normal vectors make an angle greater than a certain threshold [5].

3) Face-based method: The face clusters, which are connected sets of faces, represent the aggregate properties of the original surface at different scales rather than providing geometric approximations of varying complexity [6, 7].

Comparing these three methods, the vertex-based method and the face-based method are costly for the large-scale model, because of multiple clustering of vertices and merging of triangle meshes. In contrast, the edge-based method is the most effective way to find the feature boundaries of the large-scale model, where the model can be partitioned meaningfully. The main challenges of the edge-based method include extracting and grouping the feature edges, selecting the best-fit loop and constructing the cutting contour [8].

In this paper, we propose an improve boundary extraction method based on edge curvature estimation. First, each edge of the STL model is estimated using three curvature parameters (dihedral angle, perimeter ration and convexity). Feature edges are extracted by Genetic Algorithm (GA). Then, the discrete feature edges are grouped and filtered by Least Square Method (LSM). Finally, incomplete feature boundaries are closed by the Dijkstra's algorithm. Several experimental results are given, and the efficiency and correctness of the proposed algorithm are analysed.

2 Topology reconstruction based on Hash-table

Since the STL model has no topology information, the topology reconstruction is necessary for the boundary extraction. The vertex table of triangle facets is built to check and delete repeated vertices. As the same time, the index table of facets is constructed to save the coordinate value index of three vertices.

Set the vertex collection of a STL model is V, the edge collection is E, and the facet collection is F. According to the Euler equation, the relationship of these collections is [9]:

$$V + F - E = 2 - 2H, \tag{1}$$

where, H denotes the number of holes on the model surface. Normally, the STL model is the closed mesh surface, H=0.

***Corresponding author** e-mail: jingbinhao@cumt.edu.cn

Each facet has three edges. Meanwhile, each edge is belong to two adjacent facets. The relationship is:

$$E = 1.5F . \quad (2)$$

The Equation (1) is simplified to be:

$$V = E - F + 2 = 0.5F + 2 , \quad (3)$$

It can be seen that the data size of the vertex table is smaller than that of the facet table. The vertex collection is merged and saved in the Hash-table. Then, the vertex data of the edge collection is conversed to be the index address of the vertex Hash-table.

The common methods of Hash-table construction are [10]: the directly addressing method, the square method, the numerical analysis method, the remainder method, etc. Set the coordination value of a vertex is x, y, z , the corresponding address of Hash-table can be expressed as:

$$Index = (\text{int})((ax + by + gz)C + 0.5)\,\&T , \quad (4)$$

where, α, β, γ is the coefficients of Hash function, C is the proportionality coefficient, T is the length of Hash-table, 0.5 is the additional constant which is used to rationalise the address distribution of Hash-table. In dealing with a large data volume, the chain address method is used to handle the data conflict. Assuming the address range of Hash function is $\left[0, T-1\right]$, the pointer vector is set as *Chain-Hash[T]*. All vertices, whose address are *i*, are saved in the chain table of *Chain-Hash[i]* (As shown in Figure 1).

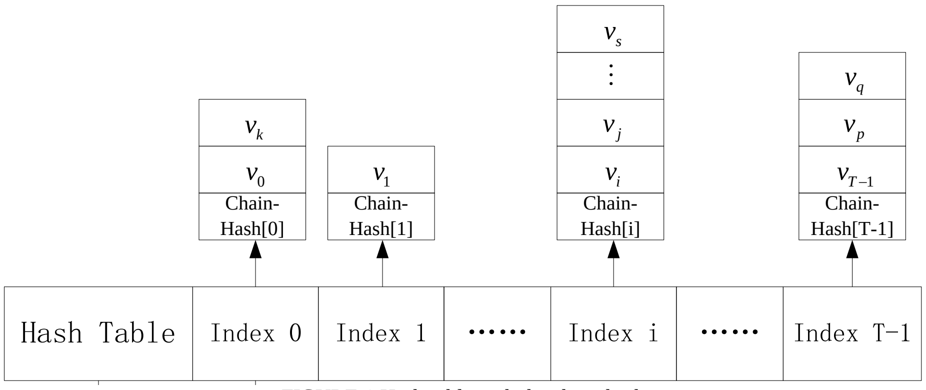

FIGURE 1 Hash-table with the chain-hash

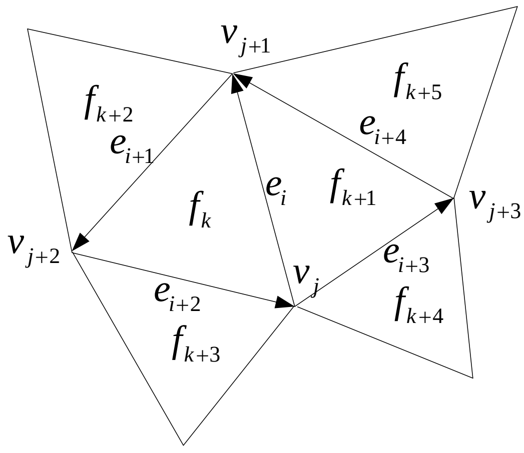

FIGURE 2 Forward-edge structure.

Considering the needs of the follow-up work, the STL model is then reconstructed based on the forward-edge structure. Set an edge e_i has four pointers: the first pointer $p_i[0]$ points to v_j , and the second pointer $p_i[1]$ points to v_{j+1} . The vector direction of $\overrightarrow{v_j v_{j+1}}$ is set as the forward direction. According to the right-hand rule, the third pointer $p_i[2]$ points to f_k , and the fourth pointer $p_i[3]$ points to f_{k+1} . Figure 2 shows the forward-edge structure of triangle facets.

3 Feature edge extracting based on GA

To extract the feature boundaries, there curvature parameters have be used to estimate the model surface.

1) Dihedral Angle: for the regular shapes and the obvious curvature changes, the feature boundaries can be directly distinguished by the dihedral angle [11].

2) Perimeter Ratio: in some regions, which have gradual curvature changes, the perimeter ratio is a sensitive criterion to find the feature boundaries.

3) Convexity: the convexity of feature boundaries is the useful parameter to determine the partition scheme[12]. Using these criteria, the feature edges with greater value than a certain threshold can be extracted.

The selection of feature thresholds is the key issue for extracting feature edges. Generally, the feature thresholds were preset based on experience or experimental analysis [13]. These traditional methods have common problems of poor universal application, low efficiency and accuracy. Both the density of facets and the curvature variation of edges should be considered when feature thresholds are selected. So we use GA to select reasonable threshold value automatically. GA is a stochastic global searching and optimizing algorithm which is suitable to solve complicated problems with large scale [14]. Figure 3 illustrates the working flow of the threshold selection based on GA.

1) Coding method: binary system is used to code individuals. The value range of the dihedral angle threshold ε is $(0,\pi/2)$. The value range of individuals is $(0,15708)\subset[0,2^{14}]$. So the size of coding scheme is 14. E.g. a binary string (10000111010010) represents an individual $x=0.8658$.

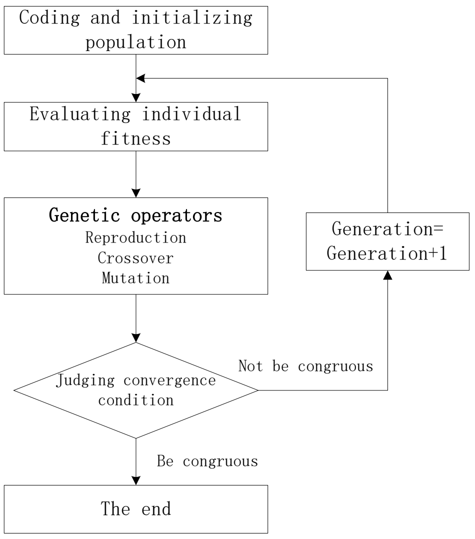

FIGURE 3 The working flow of GA

2) Initial population: the size of population is set $N=20$. First generation population $P(t)$ is created with twenty individuals A_1-A_{20}, which are initialized randomly between 0 and 15708.

3) Evaluation: in each generation, for which the GA is run, each individual in the population is evaluated against the unknown environment. The fitness values are associated with the values of objective function. We choose OTSU function to calculate the fitness values.

$$\sigma(x)^2 = w_1(x)\times w_2(x)\times(u_1(x)-u_2(x))^2, \tag{5}$$

where, $w_1(x)$ denotes the number of edges whose dihedral angle is less than x. $u_1(x)$ denotes the average dihedral angle deviation of these edges. $w_2(x)$ denotes the number of edges whose dihedral angle is greater than x. $u_2(x)$ denotes the average dihedral angle deviation of these edges whose dihedral angle is less than x. The aim is to find out the individual whose OTSU value $\sigma(x)^2$ is the greatest, which can be selected as the feature threshold.

4) Genetic operators: Genetic operators drive the evolutionary process of a population in GA, after the Darwinian principle of survival of the fittest and naturally occurring genetic operations. The most widely used genetic operators are reproduction, crossover and mutation.

The selection strategy is chiefly based on the fitness level of the individuals actually presented in the population. The sum of the individual fitness is calculated as:

$$F=\sum_{k=1}^{N}\sigma(A_k)^2 . \tag{6}$$

The selection ratio of the individual *k* is:

$$p_k=\frac{\sigma(A_k)^2}{F}, \tag{7}$$

As an individual is selected, reproduction operator only copy it from the current population into the new population without alternation.

The crossover operator starts with two selected individuals and then the crossover point (an integer between 1 and L-1, where L is the length of strings) is selected randomly. Assuming the two parental individuals are A_1 and A_2, and the crossover point is 5 (L=14). If

A_1 =(01001|101010001), A_2 =(11010|011000100)

Then the two resulting offspring are:

A_1' =(01001|011000100), A_2' =(11010|101010001)

The third genetic operator, mutation, introduces random changes in structures in the population, and it may occasionally have beneficial results: escaping from a local optimum. In our GA, mutation is just to negate every bit of the strings, i.e., changes a 1 to 0 and vice versa, with probability.

After a new population is formed by these three genetic operators, the convergence condition will be judged. If the new population is not congruous, the genetic operators of the new population will be started. Otherwise, the genetic operation is ended. The greatest individual of the new population will be selected as the feature threshold.

4 Feature edge grouping and filtering based on LSM

The extracted feature edges are discrete in the edge array, the best-fit plane (BFP) was proposed to group and link feature edges. The similar and adjacent feature edges were firstly grouped to be feature edge sets. The BFP of a feature edge set was calculated by LSM. The isolated feature edges can be grouped or deleted by BFP.

The method of LSM assumes that BFP of a feature edge set is the plane that has the minimal sum of the deviations squared (least square error) from a given set of vertices [15]. For a set of vertices in a feature edge set: $\{V_i=(x_i,y_i,z_i), i=0,1,\cdots m-1\}$, the function $S^*(V)$ represents the fitting plane that comes closest to pass through all of the vertices. According to LSM, BFP has the property that:

$$\|\delta\|=\sum_{i=0}^{m-1}\delta_i^2=\sum_{i=0}^{m-1}[S^*(V_i)-V_i]^2=\min, \tag{8}$$

for each vertex, the least square error is: $S^*(V_i)-V_i=a_0x_i+a_1y_i+a_2-z_i$, and the sum of the squares of the errors is:

$$S(a_k) = \|\delta\| = \sum_{i=0}^{m-1}[a_0 x_i + a_1 y_i + a_2 - z_i]^2 , \tag{9}$$

to make $S(a_k) = \min$, the unknown coefficients $a_k (k = 0,1,2)$ must yield zero first derivatives.

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum_{i=0}^{m-1}(a_0 x_i + a_1 y_i + a_2 - z_i)x_i = 0 \\ \dfrac{\partial S}{\partial a_1} = 2\sum_{i=0}^{m-1}(a_0 x_i + a_1 y_i + a_2 - z_i)y_i = 0 , \\ \dfrac{\partial S}{\partial a_2} = 2\sum_{i=0}^{m-1}(a_0 x_i + a_1 y_i + a_2 - z_i) = 0 \end{cases} \tag{10}$$

The conversion of Equation (10) is:

$$\begin{cases} a_0\sum x_i^2 + a_1\sum x_i y_i + a_2\sum x_i = \sum x_i z_i \\ a_0\sum x_i y_i + a_1\sum y_i^2 + a_2\sum y_i = \sum y_i z_i , \\ a_0\sum x_i + a_1\sum y_i + a_2 n = \sum z_i \end{cases} \tag{11}$$

The matrix of Equation (11) is:

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & n \end{bmatrix}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix}, \tag{12}$$

Equation (12) can be solved to quire the value of a_k, which will be used to determine BFP.

$$f_b : a_0 x + a_1 y - z + a_2 = 0 . \tag{13}$$

Some of the isolated feature edges are data noise or surface feature edges which are useless for our approach. But others may be the disconnected feature edges in certain BFP which need to be grouped for constructing feature boundaries. Checking each isolated feature edge, if this edge is almost in the nearest BFP with a tolerance, it will be marked the nearest BFP as the index. Repeat until all isolated edges have been processed. Finally, all useful adjacent and isolated feature edges have been filtered and grouped by BFP.

5 Feature boundary construction based on Dijkstra's algorithms

In most cases, the feature boundaries can be easily constructed by linking feature edges in the same BFP with the doubly linked structure which points to the neighbourhood edges. The case that requires more attention is the incomplete boundaries.

As shown in Figure 4, there are three major conditions of incomplete boundaries. For the conditions (a) and (b), the incomplete boundaries and unconnected edges are in the same BFP. Our method is to search for the shortest path of two endpoints based on the Dijkstra's algorithm, and then connect them with adding several new edges. For the condition (c), the hybrid features are in different BFPs. Our method is to close these loops separately based on the Dijkstra's algorithm. In Figure 4(c), the complete boundary belongs to four BFPs. Each incomplete loop can be closed by the above method, and then four feature loops will be obtained.

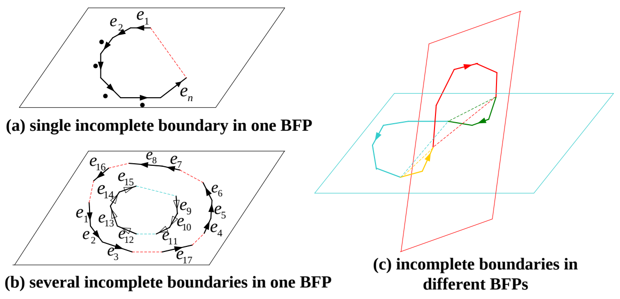
e_2
e_1
e_n
(a) single incomplete boundary in one BFP
e_{16}
e_{15}
e_8
e_7
e_6
e_{14}
e_9
e_5
e_1
e_{13}
e_{10}
e_4
e_2
e_{12}
e_{11}
e_3
e_{17}
(b) several incomplete boundaries in one BFP
(c) incomplete boundaries in different BFPs

FIGURE 4 Different conditions of incomplete boundaries

The classical Dijkstra's algorithm is the theoretical foundation for solving the problem about the shortest path. However, this algorithm is the greedy algorithm. The optimal solution can be achieved efficiently by the constraint function [16]. We use the max tolerance plane of BFP as the region constraint condition.

Set f_b^+ and f_b^- as the two max tolerance planes of BFP, the normal of f_b^+ and f_b^- is the same as that of BFP. The distance D_b^+ and D_b^- are determinate by $V_{\max}^+$ and $V_{\max}^-$, which are the farthest vertices from BFP (as shown in Figure 5).

$$f_b^+ : a_0 x + a_1 y + a_2 z + D_b^+ = 0 ,$$

$$f_b^- : a_0 x + a_1 y + a_2 z + D_b^- = 0 ,$$

where, $D_b^+ = 2 \times Dist(v_{\max}^+, f_b)$. The shortest path searching can be efficiently converged by the constraint region, which is constructed by f_b^+ and f_b^-.

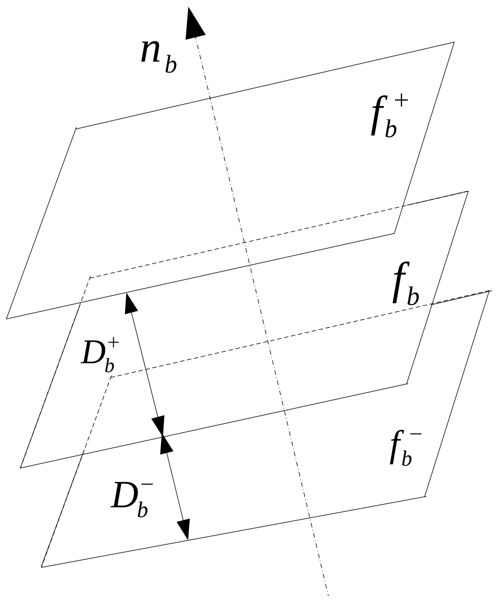

FIGURE 5 Max tolerant planes of BFP

6 Experimental results and analysis

In this section, four experimental STL models with different sizes and complexities are selected to be processed by our proposed method. The proposed algorithms have been successfully implemented in the 3DEPS system developed by the present authors in China University of Mining and Technology. A summary of results of these models is shown in Figure 6:

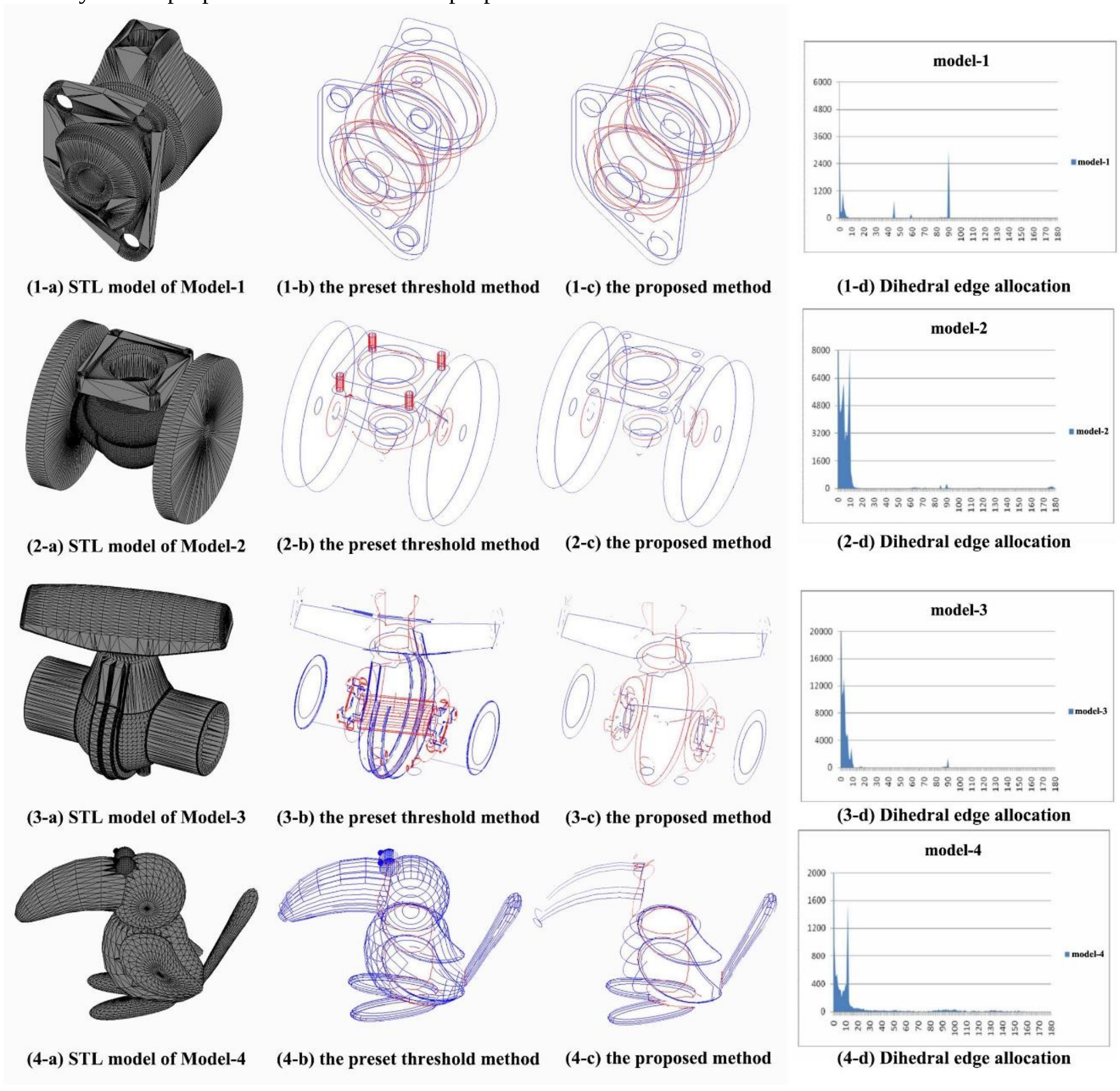
(1-a) STL model of Model-1
(1-b) the preset threshold method
(1-c) the proposed method
model-1
(1-d) Dihedral edge allocation
(2-a) STL model of Model-2
(2-b) the preset threshold method
(2-c) the proposed method
model-2
(2-d) Dihedral edge allocation
(3-a) STL model of Model-3
(3-b) the preset threshold method
(3-c) the proposed method
model-3
(3-d) Dihedral edge allocation
(4-a) STL model of Model-4
(4-b) the preset threshold method
(4-c) the proposed method
model-4
(4-d) Dihedral edge allocation

FIGURE 6 Feature boundaries extracting of experimental models

There are two types of feature boundaries:

1) the component boundary (red color ones). These kinds of boundaries are on the junction of components, which can be used to decompose the large complex model.

2) the contour boundary (blue color ones). These kinds of boundaries are on the ends of the model surface, which can be used to represent the outer contour.

For simple structure models (Model-1), the feature boundaries are less and complete. The result of the proposed method is similar to that of preset threshold method. For complex structure models (Model-2, 3, 4), the result of the proposed method is more efficient than that of preset threshold method. The amount of feature edges is about half of the preset threshold method, and useful feature edges were reserved. Take model-4 for example, the amount of feature edges that extracted by the preset threshold is 4385; the amount of feature boundaries is 136. Using the proposed method, the amount of feature edges is 1845; the amount of feature boundaries is 53. Appendix A shows the detail steps of feature edge extraction based on GA. The experimental results of other models are as shown in Table 1.

By testing the feature boundary extraction of experimental models, we can see from Figure 6 and Table 1 that our proposed method has an obvious advantage on efficiency and accuracy than the preset threshold method. Moreover, when the STL model has a large quantity of facets (more than 100,000), the computing time of our algorithm is absolutely shorter than that of vertex-based method and face-based method.

TABLE 1 Experimental results of feature boundary extraction

MODEL	Model-1	Model-2	Model-3	Model-4
Size (mm) X/Y/Z	589	723	427	402
	911	702	437	274
	775	702	263	166
Triangle	8206	57574	48362	8052
Point	4093	28777	24180	4028
Feature edge	3146	2879	4373	1845
Threshold 1 Dihedral angle	87.89°	87.28°	14.89°	15.73°
Threshold 2 Perimeter ratio	9.245	5.772	10.425	6.292
Feature boundaries	28	26	36	53

7 Conclusions

In this paper, the edge-based method is chose to extract feature boundaries. Three curvature parameters of each edge are estimated, and the feature edges with greater value than the threshold are extracted. The feature thresholds are automatically selected based on GA. Then, the discrete feature edges are grouped and filtered to be feature boundaries by BFP, which is calculated based on LSM. Finally, Dijkstra's algorithm is used to close incomplete boundaries. The extracted component boundaries can be used to decompose STL model meaningfully, and the contour boundaries can be used to represent the outer contour of STL model. The experiment results show that the proposed method has reasonable execution efficiency, which is suitable to extracting feature boundaries of large complex models. The further enhanced algorithms can be met other file formats of large complex 3D models.

Acknowledgments

The authors would like to thank the University of Nebraska-Lincoln for providing the research facility and support. This work was supported by National Natural Science Foundation of China (51305443), Natural Science Foundation of Jiangsu Province (bk20130184), and Fundamental Research Funds for the Central Universities (2012QNA27).

Appendix: A Feature edge extraction of Model-4

1) Coding method
The coding method is twelve binary code. The crossover probability p_c is set 0.6. The Mutation probability p_m is set 0.01. The stop condition is reaching the preset maximum number of iterations (the experimental number of iterations is 50 times) or the average of fitness of new generation is similar with the last generation [1,1.01].

2) Initial population

According to the coding method and population size, the initial population can be generated randomly from feasible solutions $(0, 2^{14})$. The size of initial population is $N = 20$. The randomly generated initial population $P(t)$ is as follows:

A1=(0000000 0101001)	A2=(0010101 1000111)	A3=(0110001 0111110)	A4=(1010100 0101000)
A5=(0011011 0000101)	A6=(0000000 0010000)	A7=(1011001 1010110)	A8=(1101010 1010010)
A9=(1010111 1110110)	A10=(100010 00110100)	A11=(010110 01001001)	A12=(110000 10010101)
A13=(011101 10010101)	A14=(000100 01011111)	A15=(100110 11101001)	A16=(000001 11101011)
A17=(001011 10110011)	A18=(101110 10100110)	A19=(010010 11011011)	A20=(010101 00111100)

3) Evaluation
The fitness value of an individual A_i is calculated by OTSU function $\sigma(A_i)^2$ as follows:

$\sigma(A_1)^2 =$ 4.49299e+06	$\sigma(A_2)^2 =$ 10.0996e+06	$\sigma(A_3)^2 =$ 2.47609e+06	$\sigma(A_4)^2 =$ 0.34640e+06
$\sigma(A_5)^2 =$ 8.40729e+06	$\sigma(A_6)^2 =$ 4.61561e+006	$\sigma(A_7)^2 =$ 0.88808e+06	$\sigma(A_8)^2 =$ 4.57586e+06
$\sigma(A_9)^2 =$ 0.68584e+06	$\sigma(A_{10})^2 =$ 0.11533e+06	$\sigma(A_{11})^2 =$ 3.58673e+06	$\sigma(A_{12})^2 =$ 2.12929e+06
$\sigma(A_{13})^2 =$ 0.88607e+06	$\sigma(A_{14})^2 =$ 3.98791e+06	$\sigma(A_{15})^2 =$ 0.030298e+06	$\sigma(A_{16})^2 =$ 4.42444e+06
$\sigma(A_{17})^2 =$ 9.53484e+06	$\sigma(A_{18})^2 =$ 1.40471e+06	$\sigma(A_{19})^2 =$ 5.54469e+06	$\sigma(A_{20})^2 =$ 4.20482e+06

Based on OTSU, the individual A_2 is the best, the individual A_{15} is the worst.

4) Genetic operators-roulette wheel selection

In this population, the crossover probability is as follows:

$p_1 = 0.062026$	$p_2 = 0.139426$	$p_3 = 0.034183$	$p_4 = 0.004782$
$p_5 = 0.116064$	$p_6 = 0.063719$	$p_7 = 0.012260$	$p_8 = 0.063170$
$p_9 = 0.009468$	$p_{10} = 0.001592$	$p_{11} = 0.049515$	$p_{12} = 0.029395$
$p_{13} = 0.012232$	$p_{14} = 0.055054$	$p_{15} = 0.000418$	$p_{16} = 0.061080$
$p_{17} = 0.131630$	$p_{18} = 0.019392$	$p_{19} = 0.076545$	$p_{20} = 0.058048$

After 20 times of roulette wheel selection, the new population p'_t is constructed as follows:

A'1=(0100101 1011010)	*A'2*=(0000000 0101001)	*A'3*=(0000000 0101001)	*A'4*=(0101100 1001001)
A'5=(0100101 1011010)	*A'6*=(0010111 0110010)	*A'7*=(0110001 0111110)	*A'8*=(0010101 1000111)
A'9=(0010101 1000111)	*A'10*=(010101 00111100)	*A'11*=(110101 01010010)	*A'12*=(001101 10000101)
A'13=(000001 11101011)	*A'14*=(000000 00001111)	*A'15*=(000001 11101011)	*A'16*=(001010 11000111)
A'17=(000001 11101011)	*A'18*=(010101 00111100)	*A'19*=(001101 10000101)	*A'20*=(010010 11011010)

5) Genetic operators-crossover and mutation

Since the crossover probability p_c is 0.6, N of the probabilities r_k are generated randomly from [0,1]. If $r_k < p_c$, the k chromosome of $P'(t)$ is selected. 16 individuals are chose to be paired and crossover randomly. The crossover location is selected randomly from [1,13]. The Mutation probability p_m is 0.01, and the total number of genes is 280. For each generation, 2.8 genes are mutated equally. The new population $P''(t)$ is obtained as follows,

A''_1=(010001 00101001)	A''_2=(000000 11011010)	A''_3=(000000 00101001)	A''_4=(010110 01001001)
A''_5=(010010 11011101)	A''_6=(001011 10110010)	A''_7=(011000 10100111)	A''_8=(001010 11011110)
A''_9=(000101 01010010)	A''_{10}=(010101 00111100)	A''_{11}=(111010 11000111)	A''_{12}=(001101 10000111)
A''_{13}=(000001 10101011)	A''_{14}=(000000 00001101)	A''_{15}=(000000 11000111)	A''_{16}=(001011 11101011)
A''_{17}=(000001 11101100)	A''_{18}=(010101 00111011)	A''_{19}=(001101 10000010)	A''_{20}=(010010 11011010)

The average fitness value of $P''(t)$ is 5.76753e+006, which is larger than that of $P(t)$ (3.62184e+006). Since the ratio is greater than 1.01, the next evolution is began. The fitness of A_2 is the largest in $P(t)$, while the fitness of A''_7 is the smallest in $P''(t)$. Therefore, A''_7 is replaced by A_2 to generate the next initial population $P(t+1)$. Repeat the process until the stop condition. The final population is $P(t+8)$, which is iterated eight times.

A_1=(00101010 111110)	A_2=(00111011 000101)	A_3=(11101011 011100)	A_4=(00101100 111010)
A_5=(00110110 000010)	A_6=(00110110 000010)	A_7=(00101010 111010)	A_8=(00101111 101001)
A_9=(11101010 011101)	A_{10}=(1110101 1011010)	A_{11}=(0010101 1000100)	A_{12}=(0010110 0111101)
A_{13}=(0011001 1001001)	A_{14}=(0010101 0111001)	A_{15}=(0011011 0000010)	A_{16}=(0101011 1000110)
A_{17}=(1110101 1000111)	A_{18}=(0011011 0000010)	A_{19}=(0010101 0111110)	A_{20}=(1110101 1101010)

The fitness of the best individual A_{14} is 10.165e+006. So the corresponding angle of A_{14}(15.73°) is selected as the dihedral angle threshold of Model-4.

References

[1] Kou X Y, Tan S T 2009 *Rapid Prototyping Journal* **15**(1) 5–18

[2] Luo L J, Baran I, Rusinkiewicz S, Matusik W 2012 *ACM Transactions on Graphics* **31**(6) 9–18

[3] Mao Z H, Cao G, Zhao M X 2009 *The Visual Computer* **25**(3) 289–95

[4] Razdan A, Bae M 2003 *Computer-Aided Design* **35**(9) 783–9

[5] Lien J M, Amato N M 2004 *Proceedings of the 20th Annual ACM Symposium on Computational Geometry* ACM Press New-York NY USA 17-26

[6] Lau M, Ohgawar A, Mitani J, Igrashi T 2011 *ACM Transactions on Graphics (SIGGRAPH)* **30**(4) Article No 85

[7] Hilderand K, Bickel B, Alexa M 2012 *Computer Graphics Forum (Proc. Euro graphics)* **31** 583–92

[8] Hao J, Fang L, Williams R E 2011 *Rapid Prototyping Journal* **17**(2) 116-27

[9] Lu Y, Garboczi E J 2014 *Journal of computer in civil engineering* **28**(3) 503-22

[10] Huang K, Xie G G, Li R, Xiong, S 2013 *Journal of Network and Computer Applications* **36**(2) 657-66

[11] Kadry S, Abdallah A, Joumaa C 2011 *Lecture Notes in Electrical Engineering* **133**(2) 393-7

[12] Wang M, Feng J Q, Chen W 2014 *Computers & Graphics* **38** 212-21

[13] Michikawa T, Suzuki H 2013 *Computer-Aided Design* **45**(4) 822–8

[14] Lee J. Son H, Kim C, Kim C 2013 *Automation in Construction* **35** 199–207

[15] Vigo M, Pla N, Ayala D, Martinez J 2012 *Graphical Models* **74**(3) 61-74

[16] Fu L T, Kara L B, Shimada K 2014 *Computer-Aided Design* **46** 263–8

Authors

Jingbin Hao, born in May, 1982, Xuzhou, Jiangsu Province, China

Current position, grades: PhD, Lecturer at the College of Mechanical and Electrical Engineering, China University of Mining and Technology.
University studies: PhD in Mechanical Manufacture and Automation from the University of Mining and Technology, China in 2011.
Scientific interests: rapid prototyping, laser manufacturing, computer graphics.
Publications: 15 papers.

Liang Fang, born in January, 1957, Xi'an, Shaanxi Province, China

Current position, grades: Ph. D, Professor in the State Key Laboratory of Mechanical Behaviour of Metals, Xi'an Jiaotong University, China.
University studies: PhD in Materials Science and Engineering from the Xi'an Jiaotong University, China in 1997.
Scientific interests: rapid prototyping of polystyrene, materials of tribology, advanced materials processing
Publications: over 70 papers.

Haifeng Yang, born in May, 1981, Peixian, Jiangsu Province, China

Current position, grades: PhD, associate professor at the College of Mechanical and Electrical Engineering, China University of Mining and Technology.
University studies: PhD in Materials Science from the Jiansu University, China in 2009.
Scientific interest: laser nano-manufacturing, unconventional machining technology.
Publications: 23 papers.