Polygon Crawling: Feature-Edge Extraction from a General Polygonal Surface for Mesh Generation

Soji Yamakawa[1] and Kenji Shimada[2]

The Department of Mechanical Engineering, Carnegie Mellon University
[1]soji@andrew.cmu.edu, [2]shimada@cmu.edu

Abstract.

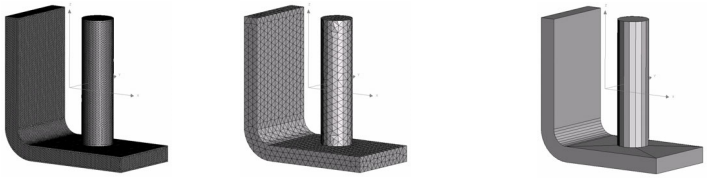
This paper describes a method for extracting feature edges of a polygonal surface for mesh generation. This method can extract feature edges from a polygonal surface typically created by a CAD facet generator in which typical feature edge extraction methods fail due to severe non-uniformity and anisotropy. The method is based on the technique called "polygon crawling," which samples a sequence of points on the polygonal surface by moving a point along the polygonal surface. Extracting appropriate feature edges is important for creating a coarse mesh without yielding self-intersections. Extensive tests have been performed with various CAD-generated facet models, and this technique has shown good performance in extracting feature edges.

1. Introduction

This paper describes a method for extracting feature edges of a polygonal surface for mesh generation; the input to the proposed method is not limited to a dense uniform triangular mesh. Feature edge extraction methods have been developed for a dense triangular mesh as shown in Fig. 1 (a), which is typically created by a laser range scanner and a uniform mesh as shown in Fig. 1 (b). The proposed method targets a general polygonal surface that includes concave and high aspect-ratio polygons. This method is particularly advantageous when feature edges must be extracted from a triangular facet generated by a CAD facet generator, which includes triangles with high aspect ratio. Existing feature edge extraction schemes do not perform well on such triangular facets. This method also tolerates certain discretization noise created by a CAD facet generator. A CAD facet generator often yields concavity where it is convex or convexity where it is concave due to a bug in the facet generator, a bad set of parameters given to the facet generator, or a deficiency of the original CAD model.

In this paper, feature edges are defined as a set of edges that must be preserved, or constrained, to obtain a valid mesh. A mesh is valid if no element

intersects with another, and if any point on the mesh is two-manifold -- except along the boundary of the open polygonal surface. The proposed method focuses on extracting the following three types of edges: (1) edges along sharp corners, (2) edges in which surface curvature, measured in the direction perpendicular to the edge, changes significantly across the edge, and (3) edges on the cylindrical surfaces.



(a) A dense triangular mesh typically created by a laser range scanner

(b) A good quality uniform mesh

(c) A general polygonal surface including concave and high aspect-ratio polygons, which the proposed method deals with

Fig. 1. Typical polygonal surfaces

Fig. 2 demonstrates the effects of feature edges on mesh generation. Fig. 2 (a) shows the input model. Fig. 2 (b) shows a meshed model without constraining edges at all, which clearly does not describe the input model well. Figures 2 (c) and 2 (d) show edges that pertain to type one and a meshed model with type one edges constrained. The mesh does not describe the fillets and the cylindrical surfaces. Figures 2 (e) and 2 (f) show edges of all three types extracted by the proposed method and a meshed model with constraining edges of all three types; of the three meshed examples it best describes the original geometry.

A simple method can detect only type one edges; it is more difficult to detect edges that pertain to types two and three. The goal of the proposed method is to detect all three types of feature edges. The proposed method can extract edges along the boundaries of any surface as type two edges -- a plane, a cylindrical surface, a spherical surface, a saddle surface, etc. -- because along such edges a sudden change of curvature occurs frequently. However, a closed (or nearly closed) cylindrical surface does not include such sudden change of curvature on the surface. The proposed method seeks out such cylindrical surfaces and extracts edges on these surfaces as type three edges.

Extracting feature edges from a polygonal surface is a much more challenging problem than extracting feature edges from a CAD model with smooth surfaces from which exact curvature at any point on the surface can be calculated. However, a polygonal surface itself does not have such information, and curvature on the surface can only be estimated. Feature edges thus can be more accurately extracted from a CAD model than from a polygonal surface. However, we often have only access to a polygonal surface for variety of reasons including incompatibility of a CAD data file between systems (we may not be able to open a received CAD file,) confidentiality requirements of the contract, or the loss of the original CAD model caused by a malfunction of a

computer or operator error. In such cases it is necessary to develop a method for estimating curvature and extracting feature edges from a polygonal surface.

The proposed method uses a procedure called polygon crawling for estimating curvature and finding feature edges. The polygon crawling method samples sequence of points, called a "crawling path," on the polygonal surface. The curvature of the polygonal surface is estimated by analyzing the crawling path. The method locates the feature edges where the estimated curvature changes significantly from one side of the edge to the other. The cylindrical surface in the polygonal surface can be found by (1) finding a principal curvature direction at a point on the polygonal surface, and (2) fitting a circle to the crawling path starting from the point toward the principal curvature direction.

The organization of the paper is as follows. Section 2 reviews previous work of the feature edge extraction. Section 3 explains the polygon crawling method. Section 4 explains the parameters that control the results of the proposed method. Section 5 describes how the curvature is estimated, and Section 6 presents the algorithm of extracting feature edges based on the curvature estimation. Section 7 shows some results, and Section 8 concludes the paper.

2. Related Work

Mangan and Whitaker present a method for partitioning a dense 3D mesh [1]. Their method calculates a curvature (or some other height function) at each vertex, and merges regions based on the curvature (or the height). Feature edges can be extracted by taking edges between partitions. Their method, however, is not designed for a mesh that includes high aspect ratio triangles.

Nomura and Hamada present a method for extracting feature edges from a triangular mesh [2]. Their method first calculates a measurement called "*concavity-convexity*," and extracts edges between a convex triangle and a concave triangle. The method also assumes a dense uniform mesh as input.

Jiao and Heath present a method for extracting feature edges [3, 4]. Their method finds a set of edges that the dihedral angle exceeds the given threshold called θ-strong edges, and takes as a feature edge a θ-strong edge that is likely to form a smooth curve when it is connected to a neighboring θ-strong edge. Their method is designed for a uniform mesh, and more work needs to be done to extend this method to a general polygonal surface.

Sun et al. present a method for detecting feature edges [5]. This method computes the likelihood of an edge to be a feature edge -- called the "edge strength," by taking vote of tensors calculated from the normal vectors of the vertices weighted by the geodesic distance from the edge. Their method is designed for segmenting a dense uniform mesh, such as a mesh created by a laser range scanner, and more work must be done to extend this method to a po-

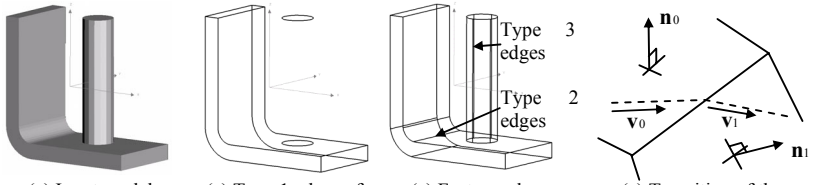
lygonal surface created by a CAD facet generator.

Baker presents a method for identifying feature edges from a triangular mesh [6]. His method uses two angle thresholds: one identifies dominant edges, and the other identifies sub-dominant edges. Dominant edges are automatically extracted as feature edges, and some sub-dominant edges are extracted based on the connection to the dominant edges. The method assumes a sufficient degree of regularity, and cannot be applied to a general polygonal surface unless some modification to the method is made.

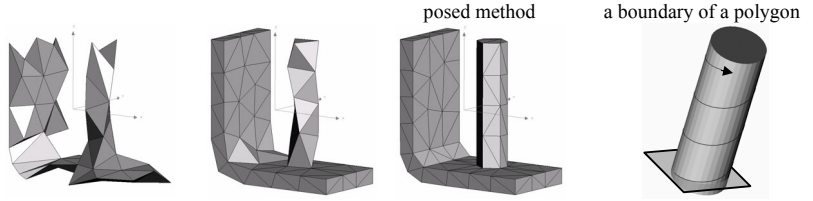
There has also been some researches on extracting feature edges from point clouds [7, 8]. These methods extract feature edges from dense sample points. These methods are not designed for taking a general polygonal surface; therefore the goal of the proposed method differs from the goals of these methods.

In summary, the existing feature edge extraction techniques for a polygonal surface can roughly be classified into two groups: (1) the techniques for a dense polygonal surface such as scanned data created by a laser range scanner, and (2) the techniques for a uniform triangular mesh. Neither type can deal effectively with a general polygonal surface that includes polygons with more than four vertices, concave polygons, and high aspect-ratio polygons.

The goal of the proposed method is to extract feature edges from a general polygonal surface, mainly a facet created by a CAD facet generator. The method tolerates a certain amount of noise imposed by a CAD facet generator. It also finds cylindrical surfaces and extracts feature edges on those cylindrical surfaces.



(a) Input model | (c) Type 1 edges of the input model | (e) Feature edges extracted by the proposed method | (a) Transition of the direction of motion across a boundary of a polygon



(b) Meshed without constraining feature edges | (d) Meshed with only type 1 edges constrained | (f) Meshed with feature edges constrained | (b) A crawling path on a discretized cylindrical surface.

Fig. 2. Effect of feature edges on mesh generation

Fig. 3. Crawling path

3. Polygon Crawling

The proposed feature edge detection technique relies on the technique called polygon crawling, which samples sequence of points along a path on the polygonal surface called a crawling path. A crawling path is analogue to a geodesic path of a smooth surface. However, a crawling path is a set of line segments as opposed to a smooth geodesic path of a smooth surface.

A crawling path is defined by an initial point on a polygonal surface, an initial direction, and termination criteria. An initial direction must be perpendicular to the normal of the polygonal surface at the initial point. Termination criteria can be anything; typically it is based on the length and/or the total angular change of the direction.

Assume that a sample point is located at the given initial point and moving toward the initial direction. The sample point maintains the same direction until it reaches the boundary of the polygon – an edge or a vertex. When the sample point moving toward $\mathbf{v}_0$ direction hits the boundary and moves out of the polygon whose unit normal is $\mathbf{n}_0$, and it moves into a polygon whose unit normal is $\mathbf{n}_1$, the moving direction changes to $\mathbf{v}_1=\mathbf{R}(\mathbf{n}_0\rightarrow\mathbf{n}_1)\mathbf{v}_0$ where $\mathbf{R}(\mathbf{n}_0\rightarrow\mathbf{n}_1)$ is a 3x3 affine transformation matrix that rotates $\mathbf{n}_0$ into $\mathbf{n}_1$ about $\mathbf{n}_0\times\mathbf{n}_1$ as shown in Fig. 3 (a). The sample point keeps moving until the termination criteria is met. A crawling path is then written as $C=\{\mathbf{p}_0,\mathbf{p}_1,...,\mathbf{p}_N\}$ where $\mathbf{p}_i$ is the point at which the sample point crosses an edge or a vertex for $(i+1)$ th time. Fig. 3 (b) shows an example of a crawling path on a discretized cylindrical surface.

We also define a "crawling plane" as $(\mathbf{x}-\mathbf{o}_{cp})\cdot(\mathbf{v}_0\times\mathbf{n}_0)=0$, where $\mathbf{x}$ is a point on the crawling plane, $\mathbf{o}_{cp}$ is the initial point of the crawling path, $\mathbf{v}_0$ is a unit vector parallel to the initial direction of the crawling path, and $\mathbf{n}_0$ is the normal of the polygon on which the initial point is lying. A crawling plane is shown in Fig. 3 (b) by the semi-transparent rectangle.

As can be seen from Fig. 3 (b), a crawling path can move away from the crawling plane. We define a "twist angle" at each point on a crawling path for measuring a directional deviation of the crawling direction from the crawling plane. If the direction of the crawling path is $\mathbf{v}_n$ ($\mathbf{v}_n$ is a unit vector) at point $\mathbf{x}$ on the crawling path, the twist angle is defined as $\theta_{tw}=\text{asin}(|\mathbf{v}_n\cdot\mathbf{n}_{cp}|)$, where $\mathbf{n}_{cp}$ is the normal of the crawling plane written as $\mathbf{n}_{cp}=(\mathbf{v}_0\times\mathbf{n}_0)$.

4. User-Specified Parameters

The result of the proposed method can be controlled by eight parameters as shown in Table 1. For obtaining the best result, these parameters must be adjusted for the meshing requirements and the behavior of the CAD facet generator. However, because it is difficult to quantify the behavior of the CAD

facet generator, a good combination of these parameters can only be found empirically. The experiments performed in this research suggest that the same set of parameters works well for the polygonal surfaces created by the same CAD facet generator.

Smaller $\omega_{shallow}$, ω_{sharp}, ω_{twist}, γ_1, and γ_2 capture more feature edges, and increases the sensitivity to the noise. Parameters ω_{cyl1} and ω_{cyl2} also control sensitivity to noise, and smaller ω_{cyl1} and ω_{cyl2} capture more cylindrical surfaces, and increases sensitivity to noise. Hence, if the CAD facet generator creates a polygonal surface with little unevenness, these parameters can be smaller, and the method will give satisfactory results. Or if the CAD facet generator creates an uneven (noisy) polygonal surface, these parameters should be increased. N_{cyl} depends on how smooth a cylindrical surface needs to be when it is meshed. Larger N_{cyl} will yield a smoother mesh on a cylindrical surface. However, it may excessively increase the number of elements. A set of parameters that gave a good performance for majority of the test models is presented in Section 7.

Table 1. Input parameters

$\omega_{shallow}$	If two vectors make an angle less than $\omega_{shallow}$, two vectors are considered to be equivalent.
ω_{sharp}	An angle threshold for finding sharp corners. It is also used for termination condition of the crawling method detailed in Section 5.2.
ω_{twist}	An angle threshold used for a termination condition of the crawling method detailed in Section 5.2.
γ_1, γ_2	Two thresholds for finding feature edges in which the curvature changes significantly, detailed in Section 6.2.
ω_{cyl1}, ω_{cyl2}	Two angle thresholds for finding cylindrical surfaces, detailed in Section 6.3.
N_{cyl}	The required number of subdivision around a complete cylinder, detailed in Section 6.3.

5. Estimating Curvature by Polygon Crawling

The proposed method estimates curvature on the polygonal surface by analyzing a crawling path. The goal is to estimate a curvature of the polygonal surface at sample point $\mathbf{x}$ measured in direction $\mathbf{v}_0$. In the following, a curvature estimation scheme of two-dimensional line segments is briefly reviewed, and it is then extended for estimating curvature of a crawling path, which is essentially the curvature estimation of the polygonal surface measured in the initial crawl direction.

5.1. Curvature Estimation of 2D Line Segments

The curvature of a curve is defined as $\kappa=1/r_c$ where r_c is the curvature radius. Hence, if two tangential vectors at the two end points of a partial circular arc of length l_{arc} makes angle θ, the curvature of the arc is calculated as $\kappa = 1/r = \theta / l_{arc}$, because $l_{arc} = r\theta$ as shown in Fig. 4 (a). Therefore, the curvature of a curve can be approximated by the angular change of a tangential vector divided by the arc length required for yielding the change.

Similarly, the curvature of line segments can be estimated by the angular change of the average tangential vector divided by the length required to yield the change. If $\mathbf{p}_0$, $\mathbf{p}_1$, …, $\mathbf{p}_N$ are the vertices of line segments, average tangential vectors are calculated as: $\mathbf{t}_0=(\mathbf{p}_1-\mathbf{p}_0)/|\mathbf{p}_1-\mathbf{p}_0|$ at $\mathbf{p}_0$, $\mathbf{t}_N=(\mathbf{p}_N-\mathbf{p}_{N-1})/|\mathbf{p}_N-\mathbf{p}_{N-1}|$ at $\mathbf{p}_N$, and $\mathbf{t}_i=(\mathbf{p}_{i+1}-\mathbf{p}_{i-1})/|\mathbf{p}_{i+1}-\mathbf{p}_{i-1}|$ at $\mathbf{p}_i$ $(0<i<N)$.

Curvature on line segments between $\mathbf{p}_i$ and $\mathbf{p}_j$ $(i<j)$ can be estimated as $\kappa(\mathbf{p}_i\rightarrow\mathbf{p}_j)=\theta_{seg}(\mathbf{p}_i\rightarrow\mathbf{p}_j)/l_{seg}(\mathbf{p}_i\rightarrow\mathbf{p}_j)$, where $\theta_{seg}(\mathbf{p}_i \rightarrow \mathbf{p}_j)$ is the angular change of the average tangential vector between $\mathbf{p}_i$ and $\mathbf{p}_j$, and $l_{seg}(\mathbf{p}_i\rightarrow\mathbf{p}_j)=\sum_{K=i}^{j-1}|\mathbf{p}_{K+1}-\mathbf{p}_K|$.

For example, when line segments are defined by six points, $\mathbf{p}_0$, $\mathbf{p}_1$, …, and $\mathbf{p}_5$, curvature of the line segments between $\mathbf{p}_1$ and $\mathbf{p}_4$ is estimated as: $\kappa(\mathbf{p}_1\rightarrow\mathbf{p}_4)=\theta_{seg}(\mathbf{p}_1\rightarrow\mathbf{p}_4)/(\sum_{K=1}^{3}|\mathbf{p}_{K+1}-\mathbf{p}_K|)$. As can be seen from Fig. 4 (b), $\sum_{K=1}^{3}|\mathbf{p}_{K+1}-\mathbf{p}_K|$ is analog to the arc length of Fig. 4 (a).

5.2. Curvature Estimation of a Crawling Path

To extend the 2D curvature estimation scheme to a crawling path, the sign of the angular change of the average tangential vector must be taken into account because the curvature must be positive if the polygonal surface is convex along the crawling path, and negative if it is concave. Let $\mathbf{p}_0$, $\mathbf{p}_1$, …, and $\mathbf{p}_N$ be points of a crawling path, $\bar{\mathbf{p}}_i$ $(0\le i\le N)$ is the projection of $\mathbf{p}_i$ on the crawling plane, and $\bar{\mathbf{t}}_0,\bar{\mathbf{t}}_1,...,\bar{\mathbf{t}}_N$ are average tangential vectors calculated from $\bar{\mathbf{p}}_0,\bar{\mathbf{p}}_1,...,\bar{\mathbf{p}}_N$. The signed angular change of the tangential vector from $\bar{\mathbf{t}}_i$ to $\bar{\mathbf{t}}_{i+1}$ is calculated as: $\theta_{seg}(\bar{\mathbf{p}}_i\rightarrow\bar{\mathbf{p}}_{i+1})=\text{acos}(\bar{\mathbf{t}}_i\cdot\bar{\mathbf{t}}_{i+1})$ if $(\mathbf{n}_0\times\mathbf{v}_0)\cdot(\bar{\mathbf{t}}_i\times\bar{\mathbf{t}}_{i+1})\ge 0$, $\theta_{seg}(\bar{\mathbf{p}}_i\rightarrow\bar{\mathbf{p}}_{i+1})=-\text{acos}(\bar{\mathbf{t}}_i\cdot\bar{\mathbf{t}}_{i+1})$ otherwise, where $\mathbf{n}_0$ is the normal vector of the polygonal surface at the initial point of the crawling path. The angular change of the tangential vector from $\bar{\mathbf{t}}_i$ to $\bar{\mathbf{t}}_j$ $(i<j)$ is calculated as: $\theta_{seg}(\bar{\mathbf{p}}_i\rightarrow\bar{\mathbf{p}}_j)=\sum_{K=i}^{j-1}\theta_{seg}(\bar{\mathbf{p}}_K\rightarrow\bar{\mathbf{p}}_{K+1})$. The curvature of the projected crawling path between $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{p}}_j$ can be estimated as: $\kappa_{seg}(\bar{\mathbf{p}}_i \rightarrow \bar{\mathbf{p}}_j) = \theta_{seg}(\bar{\mathbf{p}}_i \rightarrow \bar{\mathbf{p}}_j) / l_{seg}(\bar{\mathbf{p}}_i \rightarrow \bar{\mathbf{p}}_j)$.

For estimating the curvature of the polygonal surface at $\mathbf{x}$ in $\mathbf{v}_0$ direction, the following two issues still need to be considered: (1) termination criteria for sampling a crawling path, and (2) choice of i and j.

The proposed curvature estimation method uses three user-specified parameters, sharp angle threshold – ω_{sharp}, twist angle threshold – ω_{twist}, and shallow angle threshold – $\omega_{shallow}$ for defining the termination criteria. The proposed method samples two crawling paths $C_0 = \{\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_N\}$, and $C_1 = \{\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_M\}$. Projection of C_0 on the crawling plane is $\overline{C}_0 = \{\overline{\mathbf{a}}_0, \overline{\mathbf{a}}_1, ..., \overline{\mathbf{a}}_N\}$, and C_1 $\overline{C}_1 = \{\overline{\mathbf{b}}_0, \overline{\mathbf{b}}_1, ..., \overline{\mathbf{b}}_M\}$. The initial point of both C_0 and C_1 are $\mathbf{x}$, and the initial crawling direction of C_0 is $\mathbf{v}_0$ and of C_1 is $-\mathbf{v}_0$. For both directions, the points are sampled until: (1) total angular change of the average tangential vector on the crawling plane exceeds ω_{sharp}, i.e., both $\theta_{seg}(\overline{\mathbf{a}}_0 \to \overline{\mathbf{a}}_N)$ and $\theta_{seg}(\overline{\mathbf{b}}_0 \to \overline{\mathbf{b}}_M)$ do not exceed ω_{sharp}, (2) angular change of the normal vectors of polygons in one step of crawling exceeds ω_{sharp}, or (3) twist angle exceeds ω_{twist}.

The two crawling paths are then merged into a crawling path: $C_{all}=\{\mathbf{p}_{-(M+1)}, \mathbf{p}_{-M}, ..., \mathbf{p}_N\} = \{\mathbf{b}_M, \mathbf{b}_{M-1}, ..., \mathbf{b}_0, \mathbf{a}_0, ..., \mathbf{a}_N\}$. Projection of C_{all} on the crawling plane is $\overline{C}_{all}=\{\overline{\mathbf{p}}_{-(M+1)}, \overline{\mathbf{p}}_{-M}, ..., \overline{\mathbf{p}}_N\}$. Fig. 5 (a) shows an example of a merged crawling path.

From the merged crawling path, $\overline{C}_{all}$, $\kappa_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_j)$ can be computed for any i and j that satisfy $i < j$, $-(M+1) \le i$, $i \le N-1$, $-M \le j$, and $j \le N$, and the curvature of the polygonal surface at $\mathbf{x}$ in $\mathbf{v}_0$ direction is found by choosing appropriate i and j. However, it is very difficult to automatically choose appropriate i and j. Instead of choosing only one set of i and j, the proposed method estimates the curvature from some pairs of $l_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_j)$ and $\theta_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_j)$, or length-angle pairs, written as $\mathbf{h}_k=(l_k,\theta_k)$, by the following algorithm.

Let $k=0$
Let i be the largest integer that satisfies $\theta_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_0) \ge \omega_{shallow}$ and $i<0$.
  (If none satisfies, $i=-(M+1)$)
Let j be the smallest integer that satisfies $\theta_{seg}(\overline{\mathbf{p}}_{-1} \to \overline{\mathbf{p}}_j) \ge \omega_{shallow}$ and $0 \le j$.
  (If none satisfies, $j=N$)
Do:
  Set $\mathbf{h}_k=(l_k,\theta_k)=(l_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_j), \theta_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_j))$
  Find the next value of j, j', which satisfies:
    $\theta_{seg}(\overline{\mathbf{p}}_j \to \overline{\mathbf{p}}_{j'}) \ge \omega_{shallow}$, $j \le N$, and $j < j'$. (If none satisfies, $j'=N$)
  Find the next value of i, i', which satisfies:
    $\theta_{seg}(\overline{\mathbf{p}}_{i'} \to \overline{\mathbf{p}}_i) \ge \omega_{shallow}$, $-(M+1) < i$, and $i' < i$. (If none satisfies, $i'=-(M+1)$)
  If $l_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_0) < l_{seg}(\overline{\mathbf{p}}_{-1} \to \overline{\mathbf{p}}_j)$ replace i with i', otherwise replace j with j'.
  Increment k.
Until one of $\theta_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_j) > \omega_{sharp}$ or ($i=-(M+1)$ and $j=N$) is satisfied.

This algorithm updates i and j so that the growth of $l_{seg}(\overline{\mathbf{p}}_i \to \overline{\mathbf{p}}_0)$ and $l_{seg}(\overline{\mathbf{p}}_{-1} \to \overline{\mathbf{p}}_j)$ are balanced. Fig. 5 (b), (c), and (d) show examples of length-

angle pairs.

Finally, the curvature is estimated from the length-angle plot. On length-angle plot with length as the lateral axis and angle as the vertical axis, the slope of a function $\theta = f(l)$ describes the curvature. Hence, the curvature can be estimated by the least-square fitting of function $\theta = \kappa_{est} l$ to $\mathbf{h}_k$ as shown in Fig. 6; and κ_{est} is written as: $\kappa_{est}(\mathbf{x},\mathbf{v}_0)=\sum_k l_k\theta_k / \sum_k l_k^2$, which gives an estimated curvature at $\mathbf{x}$ measured in $\mathbf{v}_0$ direction.

5.3. Reliability of the Curvature Estimation for a Crawling Path

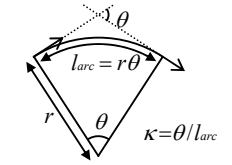
A reliability measure of the estimated curvature must be taken into account to avoid excessive extraction of feature edges. If the estimated curvature is subject to considerable error, it makes a sudden change of the estimated curvature across an edge, and the proposed method may falsely consider some edges as feature edges.

The curvature estimation scheme explained in Sections 5.1 and 5.2 becomes inaccurate where the curvature changes from negative to positive. For example, for the crawling path shown in Fig. 7 (a), the exact curvature could be zero because the sample point is on a straight line connecting two curves as shown in Fig. 7 (b), or it could be non-zero because the sample point is on one of the two curves represented by the sampled crawling path.

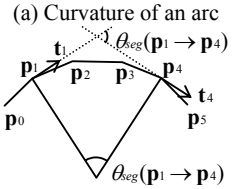
The proposed method tests the reliability of the curvature estimation as follows. Let I and J be smallest i and largest j used for sampling length-angle pairs. The curvature estimation is considered to be unreliable if all three conditions of the following are satisfied: (1) $\theta_{seg}(\mathbf{p}_I \rightarrow \mathbf{p}_0)\theta_{seg}(\mathbf{p}_{-1} \rightarrow \mathbf{p}_J)<0$, (2) $|\theta_{seg}(\mathbf{p}_0 \rightarrow \mathbf{p}_J)|>\omega_{khallow}$, and (3) $|\theta_{seg}(\mathbf{p}_i \rightarrow \mathbf{p}_0)|>\omega_{khallow}$. These conditions check whether the crawling path bends considerably to both positive and negative curvature directions. If at least one of the three conditions is not satisfied, the curvature estimation is considered to be reliable.

5.4. Finding Principal Curvature

The principal curvature at a point on the polygonal surface can be found by numerically maximizing the absolute value of the curvature estimated by the scheme explained in Section 5.2. Let $\mathbf{x}$ be a point on the polygonal surface, and $\mathbf{n}$ be the normal of the polygonal surface at $\mathbf{x}$. $\mathbf{i}$ is an arbitrary unit vector that satisfies $\mathbf{n} \cdot \mathbf{i} = 0$, and $\mathbf{j}=\mathbf{n}\times\mathbf{i}$. Any unit vector parallel to the polygon can be described as $\mathbf{v}(\phi)=\mathbf{i}\cos\phi+\mathbf{j}\sin\phi$. The problem of finding the principal curvature at $\mathbf{x}$ thus becomes a one-dimensional optimization problem: finding ϕ that maximizes $|\kappa_{est}(\mathbf{x},\mathbf{v}(\phi))|$. A conventional numerical optimization scheme such as bi-section search, Newton's method, or Powell's method, can solve this problem.

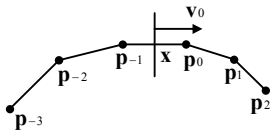


(a) Curvature of an arc

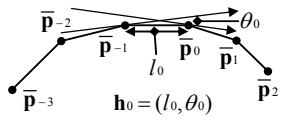


(b) Estimating the curvature of line segments

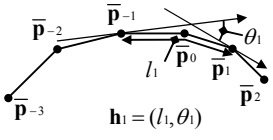
Fig. 4. Estimating curvature in 2D



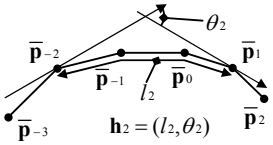
(a) Merged crawling path



(b) First length-angle pair sample ($i = -1, j = 0$)



(c) Second length-angle pair sample ($i = -1, j = 1$)



(d) Third length-angle pair sample ($i = -2, j = 1$)

Fig. 5. Crawling path C_{all} ($M = N = 2$) and length-angle pairs.

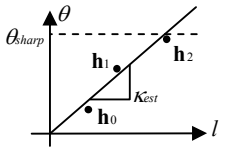
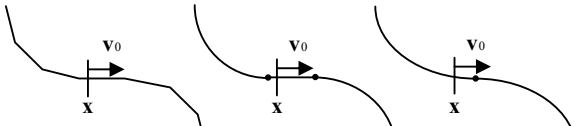


Fig. 6. Estimating the curvature by fitting a linear function to length-angle pairs



(a) Crawling Path (b) Possible curves and straight line for the crawling path (c) Possible two curves for the crawling path

Fig. 7. Ambiguity of possible curves and lines represented by the crawling path

6. Extracting Feature Edges

The proposed method extracts three different types of feature edges in the following order: (1) Feature edges on sharp corners, (2) Feature edges where the curvature changes suddenly across an edge, (3) Feature edges on the cylindrical surface.

For finding types (1) and (2) feature edges, the proposed method tests each edge. For type (3) feature edges the proposed method takes one sample point per polygon and tests whether the polygon is a part of a cylindrical surface. The total computational time for finding feature edges is thus roughly proportional to the number of edges and the number of polygons. The following section explains the methods for extracting feature edges of the three types.

6.1. Extracting Feature Edges on Sharp Corners

The proposed method first extracts feature edges on sharp corners. The pro-

posed method tests each edge of the polygonal surface shared by two polygons; if the normal vectors of the two polygons sharing the edge make a larger angle than a given angle threshold ω_{sharp}, the edge is extracted as a feature edge.

6.2. Extracting Feature Edges Based on Sudden Curvature Change

The proposed method extracts feature edges by finding a sudden change of estimated curvature across an edge. For this purpose, the curvature is measured in the direction perpendicular to the edge at two points – one point each on a polygon sharing the edge.

The proposed method visits each edge shared by two polygons and estimates the curvature for each polygon sharing the edge as follows: Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be the vertices of the edge being tested, and $\mathbf{n}$ is the normal of the polygon. The proposed method shoots two rays from $(\mathbf{e}_1+\mathbf{e}_2)/2$ to one of the directions $(\mathbf{e}_2-\mathbf{e}_1)\times\mathbf{n}$ or $(\mathbf{e}_1-\mathbf{e}_2)\times\mathbf{n}$ whichever points inside the polygon and finds the first intersection $\mathbf{e}_x$. A point inside the polygon $\mathbf{x}_{mid}$ is then found as: $\mathbf{x}_{mid}=(\mathbf{e}_x+(\mathbf{e}_1+\mathbf{e}_2)/2)/2$, as shown in Fig. 8. Since the edge is used by two polygons, two of such points, $\mathbf{x}_{mid1}$ and $\mathbf{x}_{mid2}$, are found. The proposed method then estimates curvatures at $\mathbf{x}_{mid1}$ and $\mathbf{x}_{mid2}$ as: $\kappa_{est1}=\kappa_{est}(\mathbf{x}_{mid1},\mathbf{v}_{mid1})$, and $\kappa_{est2}=\kappa_{est}(\mathbf{x}_{mid2},\mathbf{v}_{mid2})$, where $\mathbf{v}_{midK}$ is a unit vector parallel to $(\mathbf{x}_{midK}-(\mathbf{e}_1+\mathbf{e}_2)/2)$, $\{K=1,2\}$. An example of the curvature estimation at two points on the both sides of an edge is shown in Fig. 9.

Finally, if one of the following two conditions is met, the edge is extracted as a feature edge.

(1) κ_{est1} and κ_{est2} are reliable with respect to the conditions described in Section 5.3, and the ratio of $|\kappa_{est1}|$ and $|\kappa_{est2}|$ is greater than a given threshold γ_1 (i.e., $|\kappa_{est1}|>\gamma_1|\kappa_{est2}|$ or $|\kappa_{est2}|>\gamma_1|\kappa_{est1}|$)

(2) Only one of κ_{est1} and κ_{est2} is reliable with respect to the conditions explained in Section 5.3, and the ratio of $|\kappa_{est1}|$ and $|\kappa_{est2}|$ is greater than a given threshold γ_2 (i.e., $|\kappa_{est1}|>\gamma_2|\kappa_{est2}|$ or $|\kappa_{est2}|>\gamma_2|\kappa_{est1}|$)

The threshold γ_1 describes how much change of curvature across an edge is required for the edge to be extracted as a feature edge. The edge is extracted as a feature edge if the curvature changes γ_1 times across the edge. Hence, γ_1 must be chosen based on the user's requirement.

The choice of γ_2 must depend on the percentage of the error caused by the unreliability described in Section 5.3. For example, if the estimated curvature κ_{est} is subject to an error ratio of E ($E>1$), κ_{est} can vary in the range of $\kappa/E\le\kappa_{est}\le\kappa E$, where κ is the exact curvature. To take the error into account, γ_2 must be $E\gamma_1$. However, because the percentage of error E cannot be ob-

tained, γ_2 can only be found empirically.

6.3. Extracting Feature Edges on the Cylindrical Surfaces

The third procedure in the proposed method extracts feature edges on the cylindrical surfaces. It is desirable to constrain four to six edges on the cylindrical surface that are parallel to the axis of the cylinder for the meshing as shown in Fig. 2.

It is also important to constrain edges on the co-axial cylindrical surfaces consistently so that meshing does not produce self-intersections. For example, if two co-axial cylindrical surfaces have constrained edges as shown by thick lines in Fig. 10, meshing produces self-intersections when large elements need to be created for reducing the number of elements. Hence, the proposed method needs to find cylindrical surfaces from the polygonal surface, and extract feature edges on them consistently so that the chance of yielding such self-intersections is minimized.

If some polygons represent a complete cylinder (a cylinder whose cross section is a complete circle,) the cylinder can be found by fitting a circle to a crawling path in the principal curvature direction and checking the fitness of the circle to the crawling path. However, it is difficult to identify some polygons representing a partial cylinder (a cylinder whose cross section makes a partial circle) because such polygons can also be interpreted as a part of a non-cylindrical surface.

For this reason, a heuristic approach is used for finding partial cylinders. Often a complete cylinder is co-axial to a partial cylinder; when the proposed method finds some polygons that potentially represent a partial cylinder, those polygons are considered to be a part of a cylindrical surface if a complete cylinder co-axial to the partial cylinder is located close to the partial cylinder.

The proposed method uses two angle thresholds – ω_{cyl1} and ω_{cyl2} ($\omega_{cyl1} < \omega_{cyl2}$) – to find cylindrical surfaces. When the proposed method locates a set of polygons that may represent a cylindrical surface, it calculates the angle of arc of the cross section of the potential cylindrical surface, denoted as θ_{cyl}. The surface represented by the polygons is considered to be a complete cylinder if $\omega_{cyl2} \le \theta_{cyl}$. If $\omega_{cyl1} \le \theta_{cyl} < \omega_{cyl2}$, the surface represented by the polygons is considered to be a partial cylinder if a co-axial complete cylinder is found. The surface is not considered to be a part of a cylindrical surface if $\theta_{cyl} < \omega_{cyl1}$. Since θ_{cyl} is calculated by taking an angle of revolution made by a crawling path around the surface, we call θ_{cyl} a "revolution angle" of the cylinder.

After locating cylindrical surfaces represented by sets of polygons the proposed method makes groups of cylindrical surfaces based on the proximity and axial directions of the cylinders. A coordinate frame is assigned to each group, and feature edges are extracted on the cylindrical surfaces based on the coordinate frame. Details of the algorithm are described below.

The proposed method calculates a principal curvature κ_1 and the principal curvature direction $\mathbf{v}_{\kappa 1}$ at $\mathbf{x}$ by the method described in Section 5.4 where $\mathbf{x}$ is a point on the polygonal surface. The method then finds two crawling paths starting from $\mathbf{x}$ to the directions $\pm\mathbf{v}_{\kappa 1}$ until: (1) total angular change of the average tangential vector exceeds 180 degrees, (2) angular change of the normal vectors of polygons in one step of crawling exceeds ω_{sharp}, or (3) twist angle exceeds ω_{twist}.

The two crawling paths described as $\{\mathbf{a}_0,\mathbf{a}_1,...,\mathbf{a}_N\}$ and $\{\mathbf{b}_0,\mathbf{b}_1,...,\mathbf{b}_M\}$ are merged into a crawling path: $C_{cyl}=\{\mathbf{p}_{-M-1},\mathbf{p}_{-M},...,\mathbf{p}_N\}=\{\mathbf{b}_M,\mathbf{b}_{M-1},...,\mathbf{b}_0,\mathbf{a}_0,\mathbf{a}_1,...,\mathbf{a}_N\}$.

The revolution angle θ_{cyl} is calculated from C_{cyl} as: $\theta_{cyl}=\theta_{seg}(\mathbf{p}_{-M-1}\rightarrow\mathbf{p}_N)$, which is the total angular change of the tangential vector of the crawling path from $\mathbf{p}_{-M-1}$ to $\mathbf{p}_N$. If the crawling path is on a complete cylindrical surface and makes a complete revolution around the cylinder, θ_{cyl} is 360 degrees. If the crawling path is on a partial cylindrical surface, θ_{cyl} is less than 360 degrees.

If the crawling path is on a cylindrical surface, the plot of normal vectors lies on a plane, and the normal of the plane is parallel to the axial direction as shown in Fig. 11. The proposed method fits a plane to the normal vectors of the polygons that the crawling path passes by the least-square method. The plane is denoted as P_n, and the axial direction is denoted as $\mathbf{a}$.

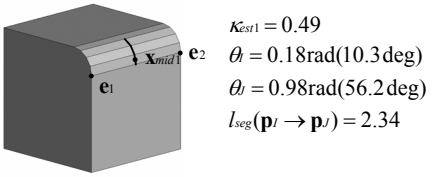
The points of the crawling path are then projected on P_n, and a circle that fits to the projected points is calculated by the least-square method [9, 10]. The center of the circle is $\mathbf{o}$, and the diameter is R. To test the fitness of the circle to the projected points, the proposed method calculates the maximum and the minimum distances from a projected point to the center of the circle, $R_{\max}$ and $R_{\min}$.

Finally, the proposed method considers that the polygons passed by the crawling path are part of a cylindrical surface if the following three conditions are satisfied: (1) $\omega_{cyl1}\le\theta_{cyl}$, (2) $0.95\le R_{\max}/R\le 1.05$, and (3) $0.95\le R_{\min}/R\le 1.05$.

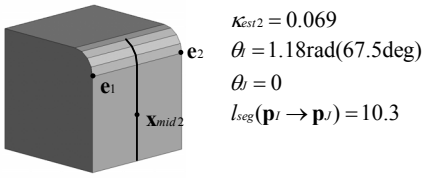
The proposed method tests the above conditions at a sample point per polygon and makes a list of detected cylindrical surfaces. If the polygon is convex, the sample point for the polygon is the center of the polygon. If the polygon is concave, the proposed method picks randomly a point inside the polygon.

Groups of cylindrical surfaces are then made by the following algorithm. Let $C_{cyl,1},C_{cyl,2},...,C_{cyl,N}$ be the detected cylinders sorted in the descending order of the revolution angle, and $\theta_{cyl,k}$ be the revolution angle of $C_{cyl,k}$ $(1\le k\le N)$; $\theta_{cyl,i}\ge\theta_{cyl,j}$ $(i<j)$. Assume that $C_{cyl,I}$ is the first cylinder that is not yet in any group and satisfies $\omega_{cyl2}\le\theta_{cyl,I}$. The proposed method makes a new group of cylindrical surfaces, $G_{cyl,K}$ (Kth group), that initially includes only $C_{cyl,I}$. It then compares $C_{cyl,I}$ with $C_{cyl,J}$ where $I+1\le J\le N$, and $C_{cyl,J}$ is added to $G_{cyl,K}$ if

(1) the axial directions of $C_{cyl,I}$ and $C_{cyl,J}$ are not different by more than $\omega_{shallow}$, and (2) $C_{cyl,I}$ and $C_{cyl,J}$ intersect each other, or one encloses the other. The two conditions checks if $C_{cyl,I}$ and $C_{cyl,J}$ are co-axial. The two conditions, however, can be satisfied if the axis of one cylinder is off center of the other. The two conditions are intentionally made loose for tolerating discretization errors. The proposed method repeats this process until all the cylinders that satisfy $\omega_{cyl2} \le \theta_{cyl,I}$ are included in a group. Some cylinders with less than ω_{cyl2} revolution angle may not belong to any group, and feature edge will not be extracted on such cylinders.



(a) Crawling path originated from $\mathbf{x}_{mid1}$



(b) Crawling path originated from $\mathbf{x}_{mid2}$

Fig. 9. Example of curvature estimation on both sides of an edge

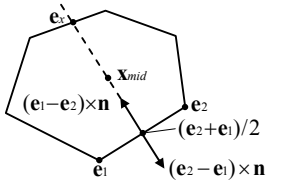
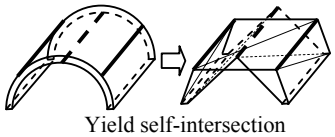


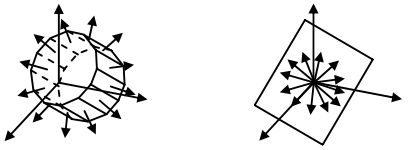
Fig. 8. Finding a point Inside a polygon



Yield self-intersection

Fig. 10. Inappropriately constrained edges

Finally, the proposed method calculates a coordinate frame for each group and makes feature edges on the surfaces of the cylinders included in the group. $\mathbf{a}_K$ is a unit vector that is parallel to the axis of the first cylinder of the group $G_{cyl,K}$, $\mathbf{i}$ is an arbitrary unit vector that is perpendicular to $\mathbf{a}_K$, and $\mathbf{j}$ is $\mathbf{i} \times \mathbf{a}_K$. The coordinate frame is defined by the three vectors $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{a}_K$. The proposed method assigns to each polygon included in the cylinders a clockwise rotation angle measured from $\mathbf{i}$ to the normal of the polygon $\mathbf{n}$, ($\theta_n = \mathbf{acos}(\mathbf{i} \cdot \mathbf{n})$ if $\mathbf{j} \cdot \mathbf{n} \ge 0$, $\theta_n = 2\pi - \mathbf{acos}(\mathbf{i} \cdot \mathbf{n})$ otherwise.) The proposed method extracts as a feature edge an edge shared by two polygons included in a cylinder of the group that satisfies $\lfloor \theta_{n1} N_{cyl}/2\pi \rfloor \ne \lfloor \theta_{n2} N_{cyl}/2\pi \rfloor$ where θ_{n1} and θ_{n2} are the angles assigned to the two polygons sharing the edge, N_{cyl} is a user-specified parameter that defines the number of feature edges needed around a complete cylinder, and $\lfloor x \rfloor$ is the largest integer that is less than x. For example, when N_{cyl} is four, $\lfloor \theta_n N_{cyl}/2\pi \rfloor$ varies zero to three, and a feature edge is extracted where $\lfloor \theta_n N_{cyl}/2\pi \rfloor$ changes as shown in Fig. 12.



(a) A discretized cylindrical surface and normal vectors

(b) A plot of normal vectors lying on a plane

Fig. 11. Normal vector plot of a discretized cylindrical surface

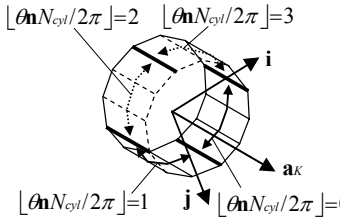


Fig. 12. Feature edges (Thick lines) around a cylinder

7. Results

The proposed method has been tested with many polygonal surfaces including more than fifty facet models used for plastic injection molding simulation, and has shown good performance. This section shows some results from the experiments. The parameters used for creating these models are shown in Table 2.

Figures 13 and 14 demonstrate the ability of the proposed method in capturing sharp corners, fillet boundaries, and cylindrical surfaces. Both models include concave polygons with more than four nodes. The proposed method successfully captures the cylindrical surfaces and extracted six feature edges on each cylindrical surface. Fig. 13 (b) shows that the proposed method extracts feature edges on the two co-axial cylinders so that a feature edges on one of the two cylinders is located close to a feature edge on the other cylinder in order to reduce the possibility of producing self-intersections when large elements must be created on the cylinders. Both models are successfully meshed into a very coarse mesh without producing self-intersections. Creating a very coarse mesh without producing self-intersections is very difficult and is only possible when appropriate edges are constrained.

Fig. 15 shows a more complex example. The input model includes concave polygons with more than four vertices. The proposed method successfully extracts feature edges from the input model, and the model is meshed into a coarse triangular mesh.

Fig. 16 shows a PDA cover. This model is created by a CAD facet generator and includes many high aspect-ratio triangles. The model has some small cylindrical holes as shown in Fig. 16 (b); such holes create self-intersections when a coarse mesh is needed unless feature edges are appropriately constrained. The proposed method successfully extracts feature edges on the holes, and a valid triangular mesh is shown in Figures 16 (e) and 16 (f).

Table 2. Parameters used for creating examples

$\omega_{shallow}$	5 degrees	ω_{twist}	45 degrees	γ_2	10	ω_{cyl1}	45 degrees
ω_{sharp}	45 degrees	γ_1	2	N_{cyl}	6	ω_{cyl2}	270 degrees

8. Discussions and Conclusions

The proposed method extracts feature edges from a general polygonal surface. The method extracts (1) edges on sharp corners, (2) edges where the curvature changes significantly across the edge, and (3) edges on the cylindrical surfaces. Such feature edges must be constrained for meshing, or some geometric features will be lost unless sufficiently small elements are used. The method has been tested with many polygonal surfaces and has shown a good performance on the test models.

The method helps create a coarse mesh without producing self-intersections. However, the method does not necessarily help create good quality elements. For creating a valid and good quality mesh, the proposed method must be used together with an appropriate element sizing function.

The method captures cylindrical surfaces and extracts edges on such cylindrical surfaces. However, the method does not capture when the cross-section is an ellipse, and/or the surface is significantly tapered. Capturing such elliptic cylinders and conic surfaces is a topic for future research.

References

[1] A. P. Mangan and R. T. Whitaker, "Partitioning 3D Surface Meshes Using Watershed Segmentation", *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, pp. 308-321, 1999.

[2] M. Nomura and N. Hamada, "Feature Edge Extraction from 3D Triangular Meshes Using a Thinning Algorithm," Proceedings of SPIE - Vision Geometry X, pp. 34-41, 2001.

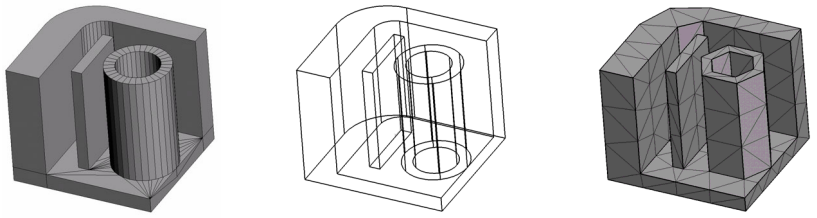
[3] X. Jiao and M. T. Heath, "Feature Detection for Surface Meshes," Proceedings of 8th International Conference on Numerical Grid Generation in Computational Field Simulations, pp. 705-714, 2002.

[4] X. Jiao and M. T. Heath, "Overlaying Surface Meshes, Part II: Topology Preservation and Feature Detection", *International Journal on Computational Geometry and Applications*, vol. 14, pp. 403-419, 2004.

[5] Y. Sun, D. L. Page, J. K. Paik, A. Koschan, and M. A. Abidi, "Triangle Mesh-Based Edge Detection and Its Application to Surface Segmentation and Adaptive Surface Smoothing," Proceedings of International Conference on Image Processing, pp. 825-828, 2002.

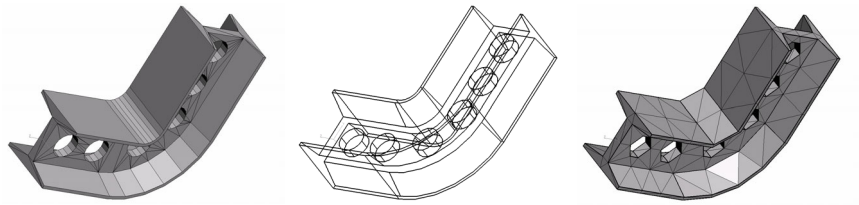
[6] T. J. Baker, "Identification and Preservation of Surface Features," Proceedings of 13th International Meshing Roundtable, pp. 299-309, 2004.

[7] S. Gumhold, X. Wang, and R. MacLeod, "Feature Extraction from Point Clouds," Proceedings of 10th International Meshing Roundtable, pp. 293-305, 2001.
[8] M. Pauly, R. Keiser, and M. Gross, "Multi-Scale Feature Extraction on Point-Sampled Surfaces," Proceedings of Eurographics, pp., 2003.
[9] W. Gander, G. H. Golub, and R. Strebel, "Least Square Fitting of Circles and Ellipses", *BIT*, vol. 34, pp. 558-578, 1994.
[10] I. D. Coope, "Circle Fitting by Linear and Nonlinear Least Squares", *Journal of Optimization Theory and Applications*, vol. 76, pp. 381-388, 1993.



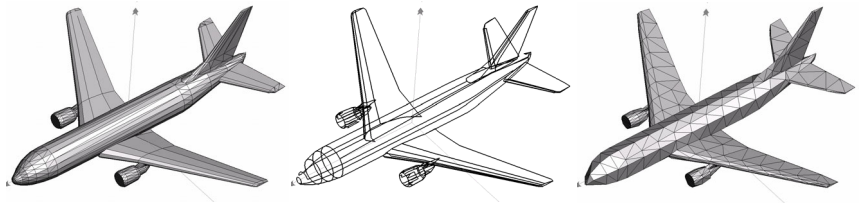
(a) Input model (235 vertices and 200 polygons) (b) Extracted feature edges (Less than 0.1 second (*)) (c) A triangular mesh (165 vertices and 326 triangles)

Fig. 13. A Mechanical Part



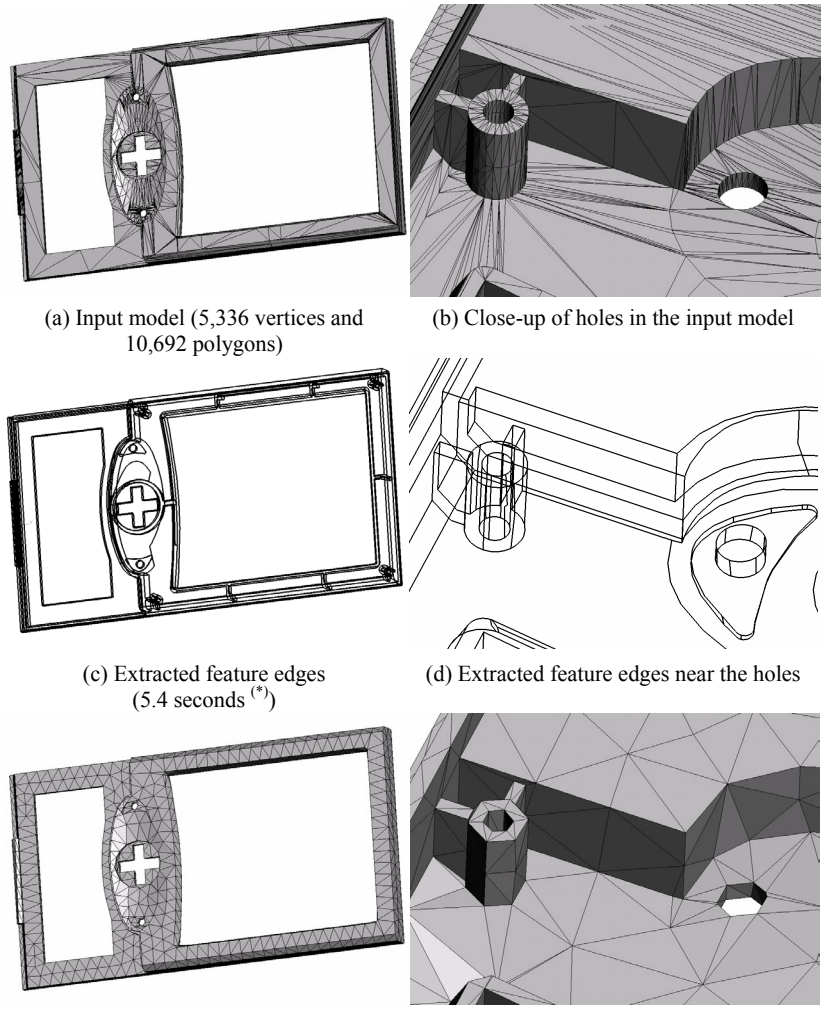
(a) Input model (356 vertices and 348 polygons) (b) Extracted feature edges (Less than 0.1 second (*)) (c) A triangular mesh (204 vertices and 428 triangles)

Fig. 14. A Bent Beam with Holes



(a) Input model (1,661 vertices and 1,082 polygons) (b) Extracted feature edges (1.3 seconds (*)) (c) A triangular mesh(444 vertices and 884 triangles)

Fig. 15. B767



(a) Input model (5,336 vertices and 10,692 polygons)

(b) Close-up of holes in the input model

(c) Extracted feature edges (5.4 seconds (*))

(d) Extracted feature edges near the holes

(e) A triangular mesh (2,007 vertices and 4,034 polygons)

(f) Close-up of the triangular mesh near the holes

Fig. 16. A front cover of PDA

(*) Computational time is measured with a PC with AMD Athlon 64 2.2GHz.