Published in Proceedings of Pacific Graphics '99, pp. 270-279.

A Discrete Spring Model for Generating Fair Curves and Surfaces

Atsushi Yamada[1], Kenji Shimada[2], Tomotake Furuhata[1], and Ko-Hsiu Hou[2]

[1]Tokyo Research Laboratory, IBM Japan Ltd., LAB-S73
623-14, Shimotsuruma, Yamato, Kanagawa, 242-8502, JAPAN
ayamada@jp.ibm.com, furuhata@jp.ibm.com

[2]Mechanical Engineering, Carnegie Mellon University
Pittsburgh, PA 15213
shimada@cmu.edu, khou+@andrew.cmu.edu

Abstract

The ability to generate fair curves and surfaces is important in computer graphics (CG), computer-aided design (CAD), and other geometric modeling applications. In this paper, we present an iteration-based algorithm for generating fair polygonal curves and surfaces that is based on a new discrete spring model. In the spring model, a linear spring, whose length approximately represents a curvature radius, is attached along the normal line of each polygon node. Energy is assigned to the difference of the lengths, that is, the difference in curvature radius, of neighboring springs. Our algorithm then minimizes the total energy by an iterative approach. It accepts as inputs (1) an initial polygonal curve (surface), which consists of a set of polygonal segments (faces) and a set of nodes as polygon-vertices, and (2) constraints for controlling the shape. The outputs are polygonal curves (surfaces) with smooth shapes. We also describe a method for improving the performance of our iterative process to obtain a linear execution time. Our algorithm provides a tool for the fair curve and surface design in an interactive environment.

Keywords: geometric modeling, fair surface design, polygonal models, energy minimization

1. Introduction

The purpose of this paper is to provide an algorithm for generating fair curves and surfaces for use in the fields of computer graphics (CG) and computer-aided design (CAD). Generation of fair shapes is a major topic in shape design [5][11][19][21][25][26][27][29]. It is also required in applications such as fitting of smooth shapes to scattered points [8][23], texture mapping [16], and so on.

The curves and surfaces treated in this paper are represented in polygonal form. The inputs of the algorithm are (1) an initial polygonal curve (surface) consisting of a set of nodes and a set of polygonal segments (faces), and (2) constraints for controlling the shape. The algorithm moves the nodes to suitable positions while minimizing the curvature variation under the given constraints. The outputs are smoothly shaped polygonal curves (surfaces). The polygonal surfaces treated in this paper are not limited to triangular meshes; they also include quadrilateral meshes. Theoretically, *n*-sided faces such as pentagons or hexagons may also be included in the meshes.

Our algorithm follows an iterative approach of the Gauss-Seidel type: node positions are iteratively updated under the given constraints. To update the node positions, two types of spring force are applied to each node: (1) a force acting in the normal direction, to optimize the curvature variation and (2) a force in the direction perpendicular to the normal, to optimize the node distribution. Then main idea of this paper lies in the discrete spring model that produces the former force minimizing curvature variation. Since curvature is a natural measurement of fairness, our spring model produces fair curves and surfaces.

As Taubin [26] points out, most energy-minimization approaches are expensive in terms of time and space. This paper also provides a method for achieving a linear execution time.

The remainder of the paper is organized as follows. In Section 2, we summarize previous work. After defining a discrete spring model in Section 3, we present a curve (surface) modeling algorithm based on the spring model in Section 4. Section 5 describes some results obtained by using the algorithm, and Section 6 summarizes the paper.

2. Previous Work

A considerable amount of work has been done on fair surface modeling. We consider that most of the previous work related to polygonal surface modeling can be classified into two types: finite-difference approaches and finite-element approaches. Our approach belongs to the former category. In both types of approach, node positions are either updated iteratively or are calculated by solving a large sparse linear system. In finite-difference approaches, values are evaluated only at nodes, while in

finite-element approaches, values are evaluated over faces by using numerical integration. One key factor that distinguishes several approaches within each category is the spring model that is used to move the node positions.

Irrespective of the classification, when the displacement of a deformation is large; that is, when the shape of the initial polygonal surface is very different from the final shape, the deformation reduces to a non-linear problem. Therefore, if solving a sparse linear system does not yield a suitable answer, the system must be solved iteratively. Large deformations appear often in the process of surface design. To construct a robust algorithm for such cases, we select an iterative approach.

The following two subsections survey both types of approach to surface modeling. In the third subsection, some other applications that use fairing as a part of their algorithm are also surveyed.

2.1 Finite-Difference Approaches to Surface Modeling

Laplacian smoothing is the easiest way to generate a fair surface. This approach iteratively moves the position of a node to the barycenter of its neighboring nodes. It has the characteristic that the area of the surface is minimized under given constraints, and is often used to improve the geometrical irregularity of a mesh in the field of finite-element meshing [12]. But when it is applied to surface modeling, the problem arises that a sharp tip is generated in the neighborhood of a node fixed by a constraint. Moreover, it is not possible to control normals by giving normal constraints.

For surface modeling, Szeliski and Tonnesen [25] used a particle system, in which each particle is defined by its position and normal, and the population of particles is controlled automatically. To obtain a fairly triangulated surface, they minimized the weighted sum of three energy factors: (1) co-planarity, (2) co-normality, and (3) co-circularity. The co-planarity condition causes neighboring nodes to lie on each other's tangent plane, the co-normality condition modifies irregular twisting of two neighboring nodes, and the co-circularity condition preserves constant curvature on edges connecting neighboring nodes. Whereas their approach uses three factors, our spring model achieves a similar effect by using just one factor.

Mallet [18][19] provided a general formulation for discrete surface interpolation with many adjustable parameters. When harmonic weighting [19] is selected among his proposed parameters - as in his results -, his approach minimizes the sum of the distances, each of which is from a node to the barycenter of its neighboring nodes.

Taubin [26] proposed a fair surface design approach in which high-frequency terms such as noise are removed by using a technique based on Fourier analysis. Due to its linear execution time, his approach is powerful for polygonal surfaces with millions of nodes, such as one captured by using a range scanner. The approach involves iterative repetition of a Gaussian operator, which moves a node to a suitable position on a line connecting a node and the barycenter of its neighboring nodes. This position is determined in such a way as to avoid shrinkage of the entire shape. Basically, Taubin's approach is suitable for removing noise from a given initial shape without causing shrinkage.

The main difference between our approach and those of Mallet and Taubin is that in our approach curvature variation is used as a measurement of minimization. Curvature is a natural measurement of fairness; consequently, our approach produces a fairer shape than theirs.

Welch and Witkin [29] also proposed a mesh-based modeling method for surfaces with arbitrary topology. After defining a local surface equation in the neighborhood of each node, which fits the latter's neighboring nodes in the least-square sense, they minimized a fairness norm based on curvature, which is calculated from the local surface. The local surface is temporarily used to evaluate the curvature at a node, and the final outputs are only node positions, not surfaces of faces. In the sense that curvature is adopted as a fairing measurement, our approach is similar to theirs. However, their approach has to solve a $5\times m$ least-square linear system (where m is the number of neighboring nodes) for each node at each iteration. This is a time-consuming process if the number of nodes is large.

Kuriyama and Tachibana [16] applied Gaussian operators not only to node positions but also to second-order derivative vectors at nodes. The magnitude of a second-order derivative vector is equal to the curvature; therefore, their approach can be interpreted as using Gaussian operators to optimize curvature. Kobbelt and his colleagues [14] also proposed a similar approach.

Eck and Jaspert [7] proposed a fairing algorithm of planar and spatial curves based on the measurement of discrete curvature and torsion. Their approach timidly moves only one node in each iteration, where the new position of the node is determined on a line connecting the node and the weighted barycenter of its neighboring nodes. The suitable position on the line is also calculated in a heuristic way.

2.2 Finite-Element Approaches to Surface Modeling

In comparison with finite-difference approaches, finite-element approaches generally yield a higher-quality surface and generate an explicit surface equation for each face, while they are

computationally more expensive.

Celniker and Gossard's approach [5] generates a C1-continuous surface by using a finite-element technique. They applied Zienkiewicz's shape function [30], which was originally proposed for finite element analysis, to attain C1 continuity in surface modeling. Their approach minimizes a weighted combination of the energy factors of a membrane and a thin plate.

Welch and Witkin [28] also proposed an approach very similar to that of Celniker and Gossard, although the former has more general formulations for both energy factors and shape control constraints.

Moreton and Sequin [21][22] presented a different type of finite-element approach in which they use biquintic Bezier patches and a fairness norm based on measures of curvature variation. Their approach produces the most impressive surface, but it is too expensive for use in an interactive environment.

2.3 Other Applications Using Fairing

Halstead and his colleagues [11] provided a solution to the problem that Catmull-Clark's subdivision surface scheme [4] gives a shrunken surface without interpolating given control points. In their process, a fairness factor proposed by Celniker and Gossard is used in conjunction with the Catmull-Clark scheme. The aim of Eck and Hoppe's work [8] is to generate tensor product surfaces from scattered points. They use a thin plate as a fairness factor to remove undulations in surfaces. Levy and Mallet [16] applied their discrete smooth interpolation approach [18][19] to the non-distorted texture-mapping problem. Koch and his colleagues [15] applied the thin-plate approach to a system for simulating facial surgery, and DeCarlo and his colleagues [6] applied the thin-plate approach to geometric modeling of human faces. As can be seen in the above literature in this subsection, fairness factors are widely used in the area of CG and CAD alongside factors unique to each study, which depend on their applications. Our approach has the potential to be applied for such applications.

3. Definition of the Discrete Spring Model: V-Spring

Figure 1 shows a planar curve and its normal lines at some sampling points on the curve. Consider the normal lines at two neighboring sampling points $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$. For a curvature-continuous curve, if $\boldsymbol{P}_i$ approaches $\boldsymbol{P}_j$, the intersection point $\boldsymbol{H}$ of the normal lines converges to a center of curvature at $\boldsymbol{P}_j$ [3].

Therefore, our idea is to attach a linear spring, as shown in Figure 2(A), to each normal line of a node consisting in a polygonal curve (surface). The linear spring works to keep equal the spring lengths $|\boldsymbol{P}_i - \boldsymbol{H}|$ and $|\boldsymbol{P}_j - \boldsymbol{H}|$ of a V-shape formed by $\boldsymbol{P}_i$, $\boldsymbol{H}$, and $\boldsymbol{P}_j$. The spring length approximately represents the curvature radius; therefore, keeping the spring length equal is equivalent to minimizing variation in the curvature radius.

Suppose node $\boldsymbol{P}_j$ is fixed by a constraint. If $|\boldsymbol{P}_i - \boldsymbol{H}|$ is smaller than $|\boldsymbol{P}_j - \boldsymbol{H}|$, as shown in Figure 2(A), node $\boldsymbol{P}_i$ moves to a new position along the normal $\boldsymbol{N}_i$ in the direction that enlarges $|\boldsymbol{P}_i - \boldsymbol{H}|$ to the size of $|\boldsymbol{P}_j - \boldsymbol{H}|$. In contrast, if $|\boldsymbol{P}_i - \boldsymbol{H}|$ is larger, node $\boldsymbol{P}_i$ moves to a new position along the normal $\boldsymbol{N}_i$ in the direction that shortens the $|\boldsymbol{P}_i - \boldsymbol{H}|$ to the size of $|\boldsymbol{P}_j - \boldsymbol{H}|$. In a stable configuration, $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ are considered to be on a circular arc whose center is at $\boldsymbol{H}$ and whose curvature radius is $|\boldsymbol{P}_j - \boldsymbol{H}|$ ($=|\boldsymbol{P}_i + \boldsymbol{dP}_i - \boldsymbol{H}|$).

Because of this V-shaped configuration of virtual springs, our spring model is named "V-Spring."

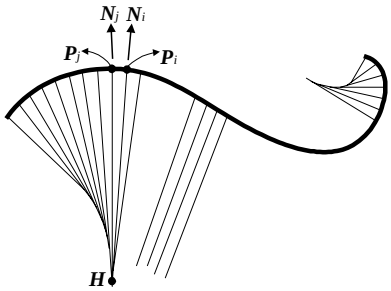

Figure 1: Planar curve and normal lines on some sampling points.

Here we formally define a displacement $\boldsymbol{dP}_{ij}$, from a current position to a new position, of a node $\boldsymbol{P}_i$ by the force of the spring model on the supposition that $\boldsymbol{P}_j$ is fixed. Let $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$ be two nodes, and let $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$, be unit normal vectors associated with the nodes, respectively. We assume the inner product dot($\boldsymbol{N}_i$, $\boldsymbol{N}_j$) to be positive; therefore, if two normal vectors with their negative inner product are given, let either $\boldsymbol{N}_i$ or $\boldsymbol{N}_j$ be the direction-reversed vector. An intersection of the two normal lines is denoted by $\boldsymbol{H}$ (see Figure 2(A)). Let t_i and t_j be real values satisfying

$$\begin{aligned} \boldsymbol{P}_i - \boldsymbol{H} &= t_i \boldsymbol{N}_i, \\ \boldsymbol{P}_j - \boldsymbol{H} &= t_j \boldsymbol{N}_j. \end{aligned} \tag{1}$$

$|t_i|$ and $|t_j|$ correspond to the distances $|\boldsymbol{P}_i - \boldsymbol{H}|$ and $|\boldsymbol{P}_j - \boldsymbol{H}|$, respectively, because $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$ are unit vectors. We define the displacement $\boldsymbol{dP}_{ij}$ of node $\boldsymbol{P}_i$ by our spring model as

$$\boldsymbol{dP}_{ij} = \left(t_j - t_i\right) \boldsymbol{N}_i. \tag{2}$$

(t_j - t_i) in the equation is calculated as follows. For the equation in which $\boldsymbol{H}$ is deleted from Equation (1), by calculating the inner products with $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$, we obtain the following equations:

$$\begin{aligned} \mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i + t_i\boldsymbol{N}_i - t_j\boldsymbol{N}_j), (\boldsymbol{N}_i)\big) = 0, \\ \mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i + t_i\boldsymbol{N}_i - t_j\boldsymbol{N}_j), (\boldsymbol{N}_j)\big) = 0. \end{aligned} \quad (3)$$

By solving Equation (3) in terms of t_i and t_j, we obtain

$$\begin{aligned} t_i &= \mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i), (-\boldsymbol{N}_i + \cos(a)\,\boldsymbol{N}_j)\big) \;/\; \big(1 - \cos^2(a)\big), \\ t_j &= \mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i), (-\cos(a)\boldsymbol{N}_i + \boldsymbol{N}_j)\big) \;/\; \big(1 - \cos^2(a)\big), \end{aligned}$$

where $\cos(a) = \mathrm{dot}(\boldsymbol{N}_i, \boldsymbol{N}_j)$. Consequently, $(t_j - t_i)$ in Equation (2) is determined as follows:

$$\begin{aligned} t_j - t_i &= \mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i), (\boldsymbol{N}_i + \boldsymbol{N}_j)\big) \;/\; \big(1 + \cos(a)\big) \\ &= \frac{\mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i), (\boldsymbol{N}_i + \boldsymbol{N}_j)\big)}{1 + \mathrm{dot}(\boldsymbol{N}_i, \boldsymbol{N}_j)}. \end{aligned} \quad (4)$$

Therefore, Equation (2) is written as

$$\boldsymbol{dP}_{ij} = \left(\frac{\mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i), (\boldsymbol{N}_i + \boldsymbol{N}_j)\big)}{1 + \mathrm{dot}(\boldsymbol{N}_i, \boldsymbol{N}_j)} \right) \boldsymbol{N}_i. \quad (5)$$

The denominator of Equation (4) always has a non-zero value, because $\cos(a)$ (=dot($\boldsymbol{N}_i$, $\boldsymbol{N}_j$,)) is assumed to be positive. One may consider from Figure 2(A) that a numerical error occurs when two normal lines are parallel because of the absence of $\boldsymbol{H}$. However, Equation (5) does not use $\boldsymbol{H}$ directly; therefore, our spring model is stable even in the case of parallel normal lines. In this case, Equation (5) is deduced to the following equation:

$$\boldsymbol{dP}_{ij} = \mathrm{dot}\big((\boldsymbol{P}_j - \boldsymbol{P}_i), \boldsymbol{N}\big)\boldsymbol{N}, \qquad (\boldsymbol{N} = \boldsymbol{N}_i = \boldsymbol{N}_j).$$

In the case of a planar curve, normal lines along $\boldsymbol{N}_i$ and $\boldsymbol{N}_j$ always have an intersection. However, if we consider a non-planar curve or a surface, normal lines do not always intersect at a point. Therefore, we cannot use Equation (1) as it is for non-planar cases. Instead of an intersection point $\boldsymbol{H}$ in Equation (1), we use $\boldsymbol{H}_i$ and $\boldsymbol{H}_j$, which are the feet of the shortest line segment connecting two normal lines (see Figure 2(B)). Then, for non-planar cases, we modify Equation (1) to obtain the following equations:

$$\begin{aligned} \boldsymbol{P}_i - \boldsymbol{H}_i &= t_i\,\boldsymbol{N}_i, \\ \boldsymbol{P}_j - \boldsymbol{H}_j &= t_j\,\boldsymbol{N}_j. \end{aligned} \quad (6)$$

For non-planar cases, the equations corresponding to Equation (3) are

$$\begin{aligned} \mathrm{dot}\big((\boldsymbol{H}_j - \boldsymbol{H}_i), (\boldsymbol{N}_i)\big) = 0, \\ \mathrm{dot}\big((\boldsymbol{H}_j - \boldsymbol{H}_i), (\boldsymbol{N}_j)\big) = 0. \end{aligned} \quad (7)$$

By solving Equation (7), we can derive an equation for $(t_j - t_i)$ in Equation (2) that is the same as Equation (4).

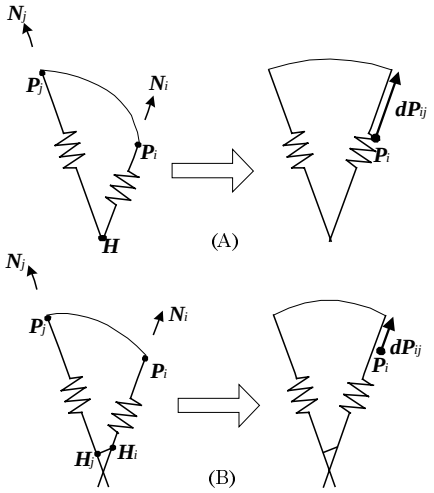

Figure 2: Definition of the spring model. (A) Displacement dP_{ij} for a planar curve. (B) Displacement dP_{ij} for a non-planar curve or a surface.

4. Curve/Surface Modeling Using a Discrete Spring Model

4.1 Overview of the Algorithm

A polygonal surface is defined as a pair of a set of nodes $\boldsymbol{P}_i$ (i = 1, ..., n), and a set of polygonal faces. We define neighboring nodes $\boldsymbol{P}_j$ (j = 1, …, m) of $\boldsymbol{P}_i$ to be a set of nodes connected to $\boldsymbol{P}_i$ by polygonal edges.

Our algorithm can be classified as an iterative method of Gauss-Seidel type. The positions and normals of nodes are updated in each iteration, and the iterations are continued until the termination condition is satisfied. In one iteration, the updated latest positions are always used to calculate the positions of other nodes. Figure 3 shows an overview of our algorithm.

```
Let Pi be the position of the i-th node
Let n be the number of nodes
While( termination condition is not satisfied ){
    For( all nodes Pi (i = 1, ..., n) ){
        Step 1: Calculate pseudo-normal Ni
    }
    For( all nodes Pi (i = 1, ..., n) ){
        Step 2: Calculate displacement dPi caused
                by the force exerted by V-Spring
        Step 3: Calculate displacement dPu, i caused
                by the force for regularizing node distribution
        Step 4: Pi = Pi + ( dPi + dPu, i )
    }
}
```

Figure 3: Overview of the algorithm.

The following subsections 4.2, 4.3, and 4.4 describe Steps 1, 2, and 3 in Figure 3, respectively. In Section 4.5 we describe several constraints. After describing the termination condition in Section 4.6, we give a method for reducing the execution time in Section 4.7. Up to Section 4.7, our discussion concerns surface modeling. Section 4.8 extends the discussion to curve modeling.

4.2 Pseudo-normal Calculation

The first step of the algorithm is to calculate a unit normal vector $\boldsymbol{N}_i$ for each node $\boldsymbol{P}_i$ of a polygonal surface (Step 1 in Figure 3). The polygonal surface is a discrete model; therefore, the unit normal vector must be calculated only approximately. In our implementation, we calculate the unit normal $\boldsymbol{N}_i$ by averaging the normals of polygonal faces adjoining the node.

There are more sophisticated ways to calculate the normal. For example, one way is to calculate the normal from a sphere fitted to the target node and its neighboring nodes in the least-square sense. However, this is more time-consuming, and fails when the target node and its neighboring nodes are on the same plane or the neighboring nodes are placed so that the target node is a saddle point. Another way, used by Taubin [26], is to calculate the normal as the average of the vectors from target node to each of its neighboring nodes. This is faster, but it fails when the target node and its neighboring nodes are on the same plane.

Our choice is the more basic but the more robust way. In the early phase of iterations, the normal vector is unreliable; however, as the iterations proceed, the normal vector converges to the reliable normal of the fair surface.

4.3 Node Displacement by the Force of V-Spring

Node $\boldsymbol{P}_i$ obtains forces from its neighboring nodes $\boldsymbol{P}_j$ (j = 1, ..., m). Each force works to keep the edge from $\boldsymbol{P}_i$ to $\boldsymbol{P}_j$ in a circular arc. Each of the displacements $\boldsymbol{dP}_{ij}$ (j = 1, ..., m) is calculated from Equation (5). Node $\boldsymbol{P}_i$ moves to a new position along the normal $\boldsymbol{N}_i$ by the weighted average $\boldsymbol{dP}_i$ of the displacements $\boldsymbol{dP}_{ij}$ (j = 1, ..., m) as follows:

$$\boldsymbol{dP}_i = \sum_{j=1}^{m} w_j \, \boldsymbol{dP}_{ij} \Big/ \sum_{j=1}^{m} w_j , \tag{8}$$

where w_j (j = 1, ..., m) are weights for the averaging. In our implementation, the weight w_j is determined by the inverse of the length of the edge connecting $\boldsymbol{P}_i$ and $\boldsymbol{P}_j$.

$$w_j = 1 \Big/ \left\| \boldsymbol{P}_i - \boldsymbol{P}_j \right\|, \quad (j = 1, 2, ..., m).$$

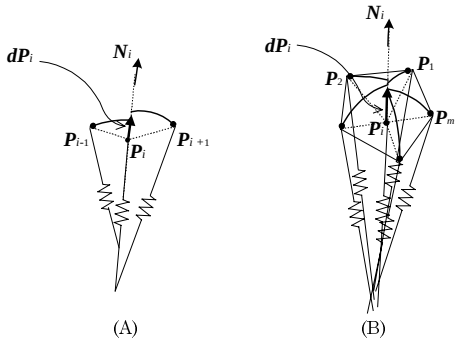

Figure 4: Node displacement by forces exerted by neighboring nodes. (A) Planar curve case. (B) Surface case.

4.4 Node Displacement for Regularizing Node Distribution

In finite-difference approaches, it is important to maintain the regular node distribution during the iteration process, because uneven distribution of nodes results in incorrect estimation of curvature. To obtain the regular node distribution, we use a variation of a Laplacian smoothing operator [12], which is a popular and effective way for removing the irregularity. The Laplacian operator moves a node to the barycenter of its neighboring nodes. However, applying the regular Laplacian operator offsets the displacement $\boldsymbol{dP}_i$ in Equation (8). Therefore, our idea is to use only a component $\boldsymbol{dP}_{u,i}$, that is perpendicular to the normal $\boldsymbol{N}_i$, of the displacement of the Laplacian operator. Using only this component creates two displacements $\boldsymbol{dP}_i$ and $\boldsymbol{dP}_{u,i}$ perpendicular to each other. Therefore, the two displacements do not offset each other. The displacement $\boldsymbol{dP}_{u,i}$ is written as follows:

$$\boldsymbol{dP}_{u,\,i} = \boldsymbol{dP}_{u0,\,i} - \boldsymbol{dP}_{u1,\,i},$$

$$\boldsymbol{dP}_{u0,\,i} = \left(\sum_{j=1}^{m} \boldsymbol{P}_j \Big/ m \right) - \boldsymbol{P}_i, \quad \boldsymbol{dP}_{u1,\,i} = \mathrm{dot}\left(\boldsymbol{dP}_{u0,\,i},\, \boldsymbol{N}_i\right) \boldsymbol{N}_i.$$

4.5 Constraints

Constraints are considered to be external forces for controlling the shape of a surface. Various kinds of constraints can be considered, depending on the requirements of applications. In this section, we describe several important constraints used in surface modeling. We classify the constraints into two types: direct and indirect constraints.

Direct Constraints

Direct constraints are given directly for a certain node. Major

constraints are:

- Positional constraints,
- Normal constraints.

A positional constraint fixes a node to a certain position during the iterations; therefore, Steps 2 and 3 in Figure 3 are skipped for the fixed node. A normal constraint fixes a normal of a node to a certain direction. For the normal fixed node, the pseudo-normal calculation (Step 1 in Figure 3) is skipped and the given normal is assigned.

In our approach, positional constraints must be given at least to end nodes for the case of an open polygonal curve and to boundary nodes for the case of an open polygonal surface. That is why the use of a Laplacian operator causes shrinkage of the shape. If it is necessary to modify the shape of the boundary curves of a surface, start from the modeling of boundary curves and go on to the modeling of the surface bounded by the boundary curves.

Indirect Constraints

Indirect constraints are not directly connected to certain nodes. During the iterations, connections between the constraints and the nodes are updated dynamically. One major constraint we introduce here is scattered points. This is an important application in CAD and CG for generating a smooth surface fitted to scattered points in the least-square sense. In the application, the constraint by a scattered point affects the point on a surface closest to it. Therefore, in our discrete model, we update the connection between the scattered point and its closest node during the iterations.

We have to modify the algorithm slightly to introduce the indirect constraints; the modified algorithm is shown in Figure 5. In Figure 5, all the steps except 2 and 5 are the same as in Figure 3.

In Step 2, each scattered point is connected to its nearest node. In practice, it is not necessary to perform Step 2 at every iteration; once in every several iterations is enough. To find the nearest node, it is enough to search neighbor of the previous nearest node in first and second orders, except when performing a first search.

In Step 5, external forces from connected scattered points $\boldsymbol{V}_j$ (j = 1, ..., m) are applied to a node $\boldsymbol{P}_i$. The displacement of $\boldsymbol{P}_i$ by $\boldsymbol{V}_j$ is calculated as follows:

$$\boldsymbol{dP}_{c,ij} = k_j \,\mathrm{dot}\left(\boldsymbol{V}_j - \boldsymbol{P}_i,\ \boldsymbol{N}_i\right) \boldsymbol{N}_i, \qquad (9)$$

where k_j denotes a weight assigned to $\boldsymbol{V}_j$. The displacement $\boldsymbol{dP}_{c,ij}$ is interpreted as a k_j -times component of vector $\boldsymbol{V}_j$ - $\boldsymbol{P}_i$ along the normal $\boldsymbol{N}_i$. In the same manner, we calculate $\boldsymbol{dP}_{c,i1}$, ..., $\boldsymbol{dP}_{c,im}$ for all connected scattered points. The final displacement of node $\boldsymbol{P}_i$ by the forces of its connected scattered points is determined by weighted-averaging of $\boldsymbol{dP}_{c,ij}$ (j = 1, ..., m), as follows:

$$\boldsymbol{dP}_{c,i} = \sum_{j=1}^{m} w_j \,\boldsymbol{dP}_{c,ij} \Big/ \sum_{j=1}^{m} w_j ,$$

where w_j (j = 1, ..., m) are weights for averaging. In our implementation, the weight w_j is determined by the inverse of the distance from node $\boldsymbol{P}_i$ to its foot $\boldsymbol{Q}_j$ to the tangent plane at $\boldsymbol{P}_i$ (see Figure 6(B)).

$$w_j = 1 \Big/ \left\| \boldsymbol{P}_i - \boldsymbol{Q}_j \right\|, \quad (j = 1, 2, ..., m).$$

The weighting means that the point closer to $\boldsymbol{P}_i$ contributes more. When $\boldsymbol{Q}_j$ is identical with or very close to $\boldsymbol{P}_i$, a sufficiently large value is assigned to its w_j to avoid division by zero.

In Equation (9), weight k_j is used to make a trade-off between fairing and keeping proximity to the scattered point $\boldsymbol{V}_j$. A larger k_j approximates $\boldsymbol{V}_j$ more closely.

In the case of a planar curve, least-square fitting is geometrically interpreted in such a way that, when a spring is attached to each line from a scattered point to the nearest point on a curve, the sum of the internal energies of the springs are minimized (see Figure 6(A)). Our approach is geometrically interpreted as being to attach a spring to a line from a scattered point to its foot in the tangent plane at the nearest node (see Figure 6(B)). We therefore consider that our approach is approximately equivalent to least-square fitting.

The advantage of our approach is that it can be applied not only to surfaces with regular topology, such as tensor product surfaces, but also to surfaces with arbitrary topology. In addition, theoretically, n-sided polygons such as pentagons or hexagons may be included in the polygonal surface.

```
Let Pi be the position of the i-th node
Let n be the number of nodes
Let Vi be the i-th scattered point
Let l be the number of scattered points
While( termination condition is not satisfied ){
    For( all nodes Pi (i = 1, ..., n) ){
        Step 1: Calculate pseudo-normal Ni
    }
    For( all scattered points Vi (i = 1, ..., l) ){
        Step 2: Make connection of each scattered point
                to its nearest node
    }
    For( all nodes Pi (i = 1, ..., n) ){
        Step 3: Calculate displacement dPi caused
                by the force exerted by V-Spring
        Step 4: Calculate displacement dPu, i caused
                by the force for regularizing node distribution
        Step 5: Calculate displacement dPc, i  caused
                by the force exerted by scattered points
        Step 6: Pi = Pi + (dPi + dPu, i + dPc, i)
    }
}
```

Figure 5: Overview of the fitting algorithm.

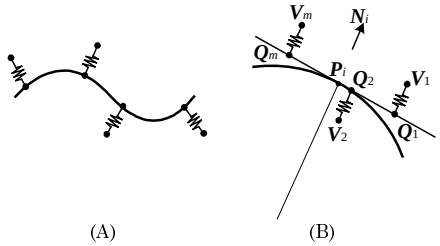

Figure 6: Least-square fitting of a planar curve to scattered points. (A) Geometric interpretation of general least-square fitting. (B) Geometric interpretation of our fitting approach.

4.6 Termination Condition

To determine when to terminate iterations, the maximum among the norms of all node displacements is compared with a given threshold *e*. If the maximum norm is less than the threshold *e*, the iterations are terminated. The size of the displacement depends on the resolution of the polygonal surface; therefore the displacement should be normalized by the sizes of polygonal faces. In our implementation, we normalize the displacement of each node by the average length of its neighboring edges. The maximum norm of the normalized displacements is then compared with the threshold *e*.

4.7 Performance Improvement

Iterative approaches of the Jacobi or Gauss-Seidel types tend to remove high frequencies quickly, while they take many iterations to remove low frequencies. Consequently, if the number of nodes is extremely large, it takes many iterations to achieve convergence. A promising way to reduce the execution time is to employ multi-grid methods [2][10][20]. Kobbelt and his colleagues [14] are positively using the multi-grid method to model dense meshes. By using the multi-grid method, a linear execution time can be achieved.

The multi-grid method requires polygonal surfaces with several different levels of resolutions. Such surfaces can be obtained by the use of mesh simplification algorithms [1][9][13][24][27]. For example, let *M*1, *M*2, and *M*3 be three levels of polygonal surfaces, where *M*1 is the finest and *M*3 is the coarsest (see Figure 7). The V-cycle multi-grid method applies iterations for polygonal surfaces with different levels, in the sequence {*M*1, *M*2, *M*3, *M*2, *M*1}. The first half of the process, going down from *M*1 to *M*2, is called pre-smoothing, and the second half of the process, coming up from *M*3 to *M*1, is called post-smoothing. In the pre-smoothing, some iterations are performed at each level in order to remove noise. On the coarsest level *M*3, a rough shape is predicted by the solution. As the post-smoothing proceeds, the rough shape approaches the precise shape.

In the post-smoothing, iterations are performed at each level until the termination condition described in Section 4.6 is satisfied. Our termination condition is normalized by the resolution; this provides an efficient way of determining the time at which to move on.

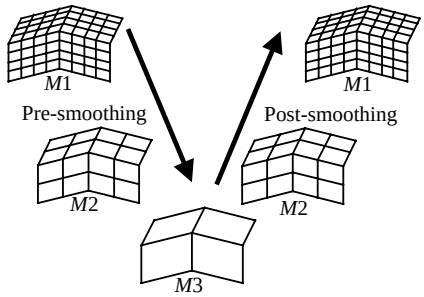

Figure 7: Concept of the multi-grid method

4.8 Extension to Curve Modeling

Basically, the algorithm for curve fairing is the same as the one for surface fairing. In each iteration, the node $\boldsymbol{P}_i$ is moved to a new position by the forces exerted by two neighboring nodes $\boldsymbol{P}_{i-1}$ and $\boldsymbol{P}_{i+1}$ (see Figure 4(A)).

The case of a planar curve does not involve any extension of the surface case; however, in the case of a non-planar curve, the calculation of the pseudo-normal $\boldsymbol{N}_i$ is more difficult than in the surface case. From a sequence of nodes $\boldsymbol{P}_{i-1}$, $\boldsymbol{P}_i$, and $\boldsymbol{P}_{i+1}$, we calculate the unit tangent $\boldsymbol{T}_i$ and the unit binormal $\boldsymbol{B}_i$ as follows:

$$\boldsymbol{T}_i = (\boldsymbol{P}_{i+1} - \boldsymbol{P}_{i-1}) \,/\, \|\boldsymbol{P}_{i+1} - \boldsymbol{P}_{i-1}\|,$$
$$\boldsymbol{B}_i = (\boldsymbol{P}_i - \boldsymbol{P}_{i-1}) \times (\boldsymbol{P}_{i+1} - \boldsymbol{P}_i) \,/\, \| (\boldsymbol{P}_i - \boldsymbol{P}_{i-1}) \times (\boldsymbol{P}_{i+1} - \boldsymbol{P}_i) \|,$$

where $\times$ denotes the outer product. As the outer product of $\boldsymbol{B}_i$ and $\boldsymbol{T}_i$, we calculate the unit principal normal $\boldsymbol{N}_i$ as follows:

$$\boldsymbol{N}_i = \boldsymbol{B}_i \times \boldsymbol{T}_i \,/\, \| \boldsymbol{B}_i \times \boldsymbol{T}_i \|.$$

If $\boldsymbol{P}_{i-1}$, $\boldsymbol{P}_i$, and $\boldsymbol{P}_{i+1}$ are collinear, $\boldsymbol{B}_i$ and $\boldsymbol{N}_i$ are zero vectors; then the displacement $\boldsymbol{dP}_i$ is a zero vector according to Equation (5). The best way to obtain a fair curve is to apply V-Spring forces in the directions of both $\boldsymbol{B}_i$ and $\boldsymbol{N}_i$; however, in practice, applying a force only in the direction of $\boldsymbol{N}_i$ gives a fair curve, even if the problem is a non-planar case.

5. Results

Figure 8 shows a result of fair surface generation obtained by using our algorithm. Figure 8(A) shows the initial mesh with sharp corners, that is, sudden changes in curvature, while figures (B) and

(C) show the faired results with two different sets of direct constraints. As can be seen in the figures, the algorithm produced fairer surfaces. The surface of Figure 8(B) resulted when the positions of nodes on the inner and outer boundaries were kept unchanged by using the positional constraints. The surface of Figure 8(C) was generated by constraining the normals of the nodes on the inner and outer boundaries, in addition to the positions of these nodes. Figure 10 shows another result of fairing in a shaded image.

Figure 9 shows the stability of our algorithm by giving it an extremely noisy mesh as initial data. We added random noise of large amplitude to the mesh in Figure 8(A), and then processed the noisy mesh by using our algorithm. Five iterations of pre-smoothing generated the smoother surface shown in Figure 9(B). With further processing, the mesh converged to the fair surface shown in Figure 9(C).

Table 1 shows the execution time of our algorithm measured for meshes of various resolutions. It indicates that the time grows approximately linearly to the complexity of the given meshes.

Table 1: Execution time for fair surface generation. The execution time is measured for the surface shape with the constraints shown in Figure 8(C). Column (a) in the table contains data for the resolution shown in Figure 8(C). Columns (b), (c), and (d) contain data for different resolutions with the same surface shape. These data are measured under the following conditions:

CPU: Pentium II 450 MHz,
System: Windows NT 4.0,
Threshold e for termination condition: 0.001,
Number of levels for multi-grid: 6,
Number of iterations at each level of pre-smoothing: 5.

	(a)	(b)	(c)	(d)
Number of nodes	553	1610	3644	14962
Number of faces (triangles)	985	3018	6983	29316
Execution time (sec)	2.38	13.8	36.4	177.1

Figure 11 shows the results of least-square fitting of polygonal curves, using indirect constraints as described in Section 4.5. The initial curve, shown by the noisy thin line, converged to a smooth curve, shown by the thick line. The node positions of the initial curve are assigned to scattered points. In Figure 11(A), all the weights k_i in Equation (9) that are used to realize a trade-off between fairing and keeping proximity are set to 0.01, while in Figure 11(B) they are set to 0.00001. As seen in the figures, the algorithm produced fairer curves under the indirect constraints.

Figure 12 shows the result of least-square fitting of a polygonal surface to scattered points. Even though one pentagonal face is included in the polygonal surface, a smooth profile is generated and fitted to the scattered points.

6. Summary

This paper has presented an algorithm for generating fair polygonal curves and surfaces based on an iterative approach, using a new discrete spring model. The algorithm produced polygonal curves and surfaces whose local variation in curvature is minimized.

In our discrete spring model, a linear spring, whose length approximately represents a curvature radius, is attached along the normal line of each node. Energy is assigned to the difference of the lengths, that is, the difference in the curvature radius, of nearby springs. Our algorithm then tries to minimize the total energy by an iterative approach. It accepts various constraints, such as positional, normal, and least-square constraints, so that useful surface models can be generated for computer graphics, computer aided design, and other geometric modeling applications.

The implementation of the algorithm is easy on account of its geometrically intuitive interpretation. The results of experiments showed that the algorithm generated fair surfaces that satisfy positional, normal, and other constraints. The algorithm was robust to positional noise in the initial polygonal data. Computational costs increased approximately linearly to the number of nodes in a polygonal curve (surface).

In our experience, the algorithm is decidedly robust and stable; however, the convergence of the algorithm need to be proved in future work. The objective of this paper is to apply our spring model to shape modeling; however, the fairing problem is not limited to this application and we will apply our spring model to other applications in the course of our future work.

References

[1] N. Amenta, M. Bern, and M. Kamvysselis, A New Voronoi-Based Surface Reconstruction Algorithm, Computer Graphics (SIGGRAPH'98 Conference Proceedings), pp. 415-421, 1998.

[2] W. Briggs, A Multi-grid Tutorial, SIAM, Philadelphia, 1977.

[3] M. P. Do Carmo, Differential Geometry of Curves and Surfaces, Prentice Hall, 1976.

[4] E. Catmull and J. Clark, Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes, Computer Aided Design, 10(6), pp. 350-355, 1978.

[5] G. Celniker and D. Gossard, Deformable Curve and Surface Finite-Elements for Free-Form Shape Design, Computer Graphics (SIGGRAPH'91 Conference Proceedings), pp. 257-266, 1991.

[6] D. DeCarlo, D. Metaxas, and M. Stone, An Anthropometric Face Model using Variational Techniques, Computer Graphics (SIGGRAPH '98 Conference Proceedings), pp. 67-74, 1998.

[7] M. Eck and R. Jaspert, Automatic Fairing of Point Sets, in "Designing Fair Curves and Surfaces," N. S. Sapidis Eds, SIAM, pp. 45-60, 1994.

[8] M. Eck and H. Hoppe, Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type, Computer Graphics (SIGGRAPH '96 Conference Proceedings), pp. 325-334, 1996.

[9] M. Garland and P. S. Heckbert, Surface Simplification Using Quadric Error

Metrics, Computer Graphics (SIGGRAPH '97 Conference Proceedings), pp. 209-216, 1997.

[10] W. Hackbusch, Multi-Grid Methods and Applications, Springer-Verlag, Berlin, New York, 1985.

[11] M. Halstead, M. Kass, and T. DeRose, Efficient, Fair Interpolation using Catmull-Clark Surfaces, Computer Graphics (SIGGRAPH '93 Conference Proceedings), pp. 35-44, 1993.

[12] K. Ho-Le, Finite Element Mesh Generation Methods: A Review and Classification, Computer Aided Design, 20(1), pp. 27-38, 1988.

[13] H. Hoppe, Progressive Meshes, Computer Graphics (SIGGRAPH '96 Conference Proceedings), pp. 99-108, 1996.

[14] L. Kobbelt, S. Campagna, J. Vorsatz, and H. P. Seidel, Interactive Multi-Resolution Modeling on Arbitrary Meshes, Computer Graphics (SIGGRAPH '98 Conference Proceedings), pp. 105-114, 1998.

[15] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Buren, G. Fankhauser, and Y. I. H. Parish, Simulating Facial Surgery Using Finite Element Methods, Computer Graphics (SIGGRAPH '96 Conference Proceedings), pp. 421-428, 1996.

[16] S. Kuriyama and K. Tachibana, Polyhedra Surface Modeling with a Diffusion System, Computer Graphics Forum, Vol. 16, No. 3, pp. 39-46, 1997.

[17] B. Levy and J. L. Mallet, Non-Distorted Texture Mapping for Sheared Triangulated Meshes, Computer Graphics (SIGGRAPH '98 Conference Proceedings), pp. 343-352, 1998.

[18] J. L. Mallet, Discrete Smooth Interpolation, ACM Transactions on Graphics, 8(2), pp121-144, 1989.

[19] J. L. Mallet, Discrete Smooth Interpolation in Geometric Modelling, Computer Aided Design 24(4), 1992, pp. 178-191.

[20] C. McCormick, Multilevel Adaptive Methods for Partial Differential Equations, SIAM, Philadelphia, 1989.

[21] H. P. Moreton and C. H. Sequin, Functional Optimization for Fair Surface Design, Computer Graphics (SIGGRAPH '92 Conference Proceedings), pp. 167-176, 1992.

[22] H. P. Moreton and C. H. Sequin, Minimum Variation Curves and Surfaces for Computer-Aided Geometric Design, in "Designing Fair Curves and Surfaces," N. S. Sapidis Eds, SIAM, pp. 123-159, 1994.

[23] D. F. Rogers and N. G. Fog, Constrained B-Spline Curve and Surface Fitting, Computer Aided Design, 21(10), pp. 641-648, 1989.

[24] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, Decimation of Triangle Meshes, Computer Graphics (SIGGRAPH '92 Conference Proceedings), pp. 65-70, 1992.

[25] R. Szeliski and D. Tonnesen, Surface Modeling with Oriented Particle Systems, Computer Graphics (SIGGRAPH '92 Conference Proceedings), pp. 185-194, 1992.

[26] G. Taubin, A Signal Processing Approach to Fair Surface Design, Computer Graphics (SIGGRAPH '95 Conference Proceedings), pp. 351-358, 1995.

[27] G. Turk, Re-Tiling Polygonal Surfaces, Computer Graphics (SIGGRAPH '92 Conference Proceedings), pp. 55-64, 1992.

[28] W. Welch and A. Witkin, Variational Surface Modeling, Computer Graphics (SIGGRAPH '92 Conference Proceedings), pp. 157-166, 1992.

[29] W. Welch and A. Witkin, Free-Form Shape Design Using Triangulated Surfaces, Computer Graphics (SIGGRAPH '94 Conference Proceedings), pp. 247-256, 1994.

[30] O. C. Zienkiewicz, The Finite Element Method, Third Edition, McGraw-Hill, United Kingdom, 1977.

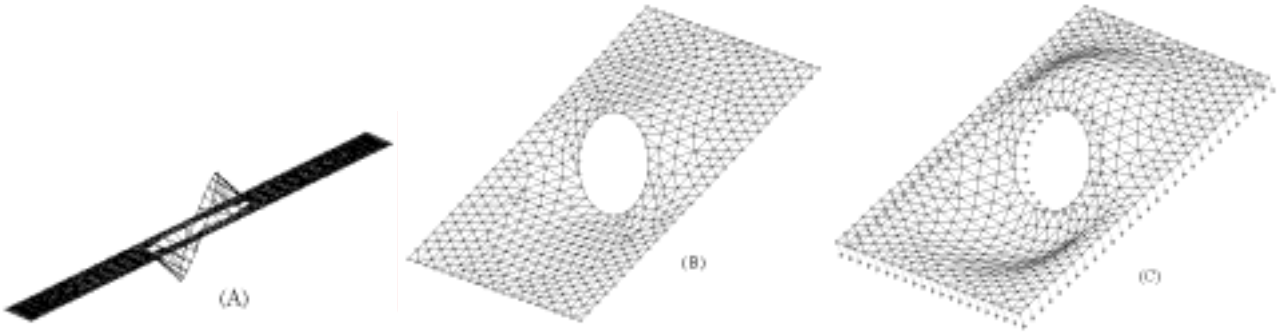

Figure 8: (A) Initial mesh. (B) Deformed result from (A) under the positional constraints on inner- and outer-boundary nodes. (C) Deformed result from (A) under the positional and normal constraints on inner- and outer-boundary nodes. Arrows on boundary nodes show the directions of normal constraints.

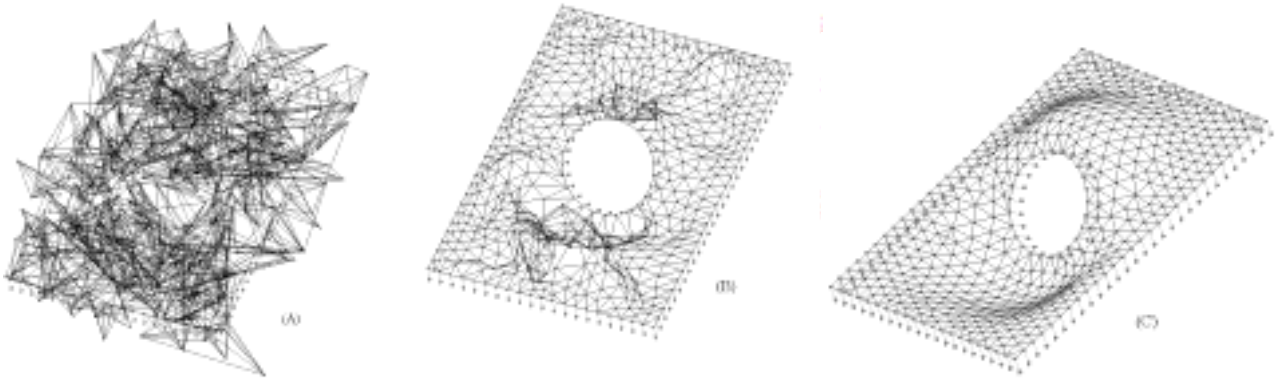

Figure 9: Stability of the algorithm. (A) Initial mesh with random noise on the nodes. (B) Shape (A) after five iterations of pre-smoothing on the finest level. (C) Shape (A) after full convergence.

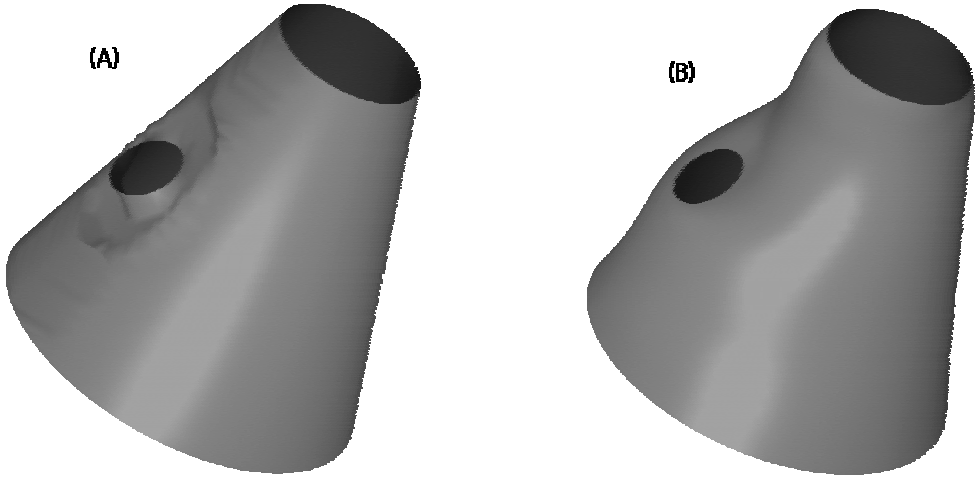

Figure 10: Shaded image of a deformed result. (A) Initial mesh. (B) Deformed result from (A) under the positional and normal constraints on nodes of upper-, lower-, and inner-boundaries.

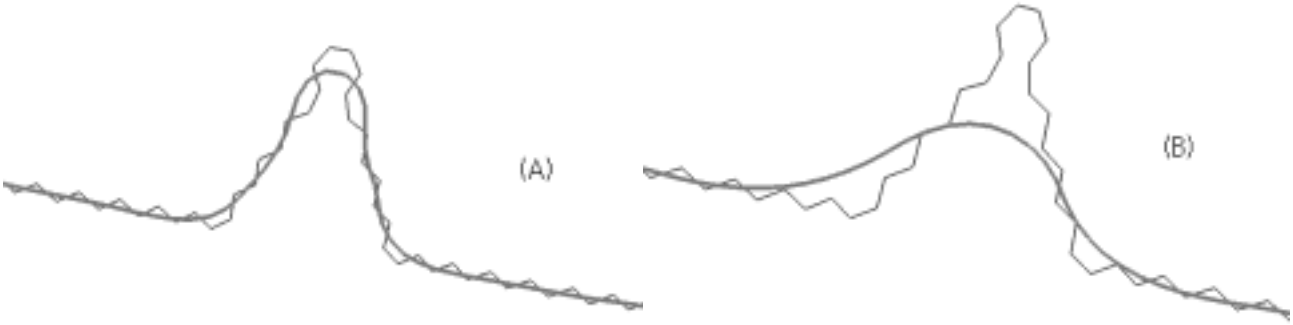

Figure 11: Least-square fitting of a polygonal curve to scattered points. A noisy thin line represents an initial curve. A smooth thick line represents a converged curve. Node positions of the initial curve are given as scattered points. (A) All weights k_j in Equation (9) are set to 0.01. (B) All weights k_j are set to 0.00001.

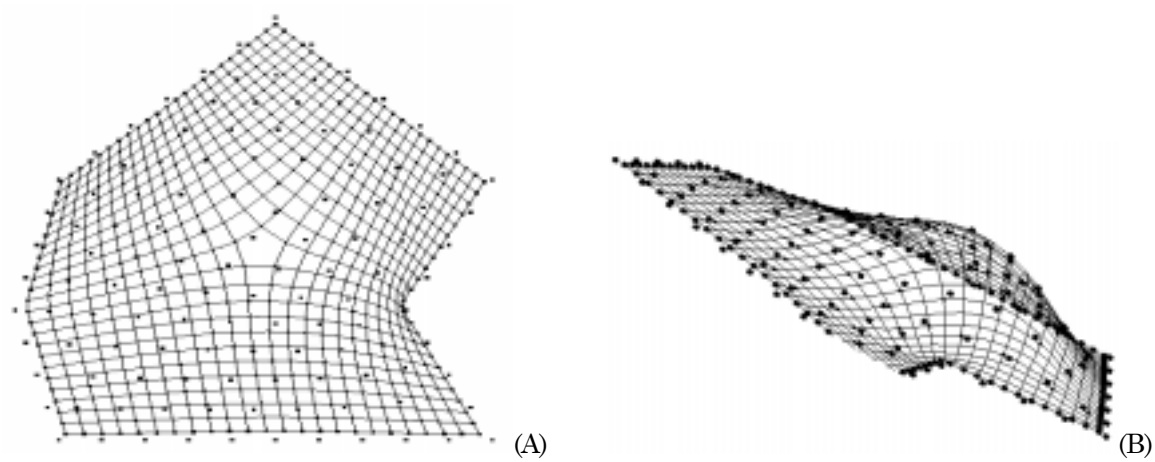

Figure 12: Least-square fitting of a polygonal surface including a pentagonal face to scattered points. (A) and (B) are different view of the same surface.