Optimal Surface Flattening

Danny Z. Chen[1,⋆] and Ewa Misiołek[2]

[1] Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA
chen@cse.nd.edu

[2] Mathematics Department, Saint Mary's College, Notre Dame, IN 46556, USA
misiolek@saintmarys.edu

Abstract. The problem of optimal surface flattening in 3-D finds many applications in engineering and manufacturing. However, previous algorithms for this problem are all heuristics without any quality guarantee and the computational complexity of the problem was not well understood. In this paper, we prove that the optimal surface flattening problem is NP-hard. Further, we show that the problem admits a PTAS and can be solved by a $(1+\epsilon)$-approximation algorithm in $O(n \log n)$ time for any constant $\epsilon > 0$, where n is the input size of the problem.

1 Introduction

We consider the problem of "flattening" a surface S in $\mathbb{R}^3$. Surface flattening is an important problem that finds applications in computer graphics and surface reconstruction as well as in engineering and manufacturing [3] (particularly in aircraft, vehicle, or garment design). In the latter applications, the surface of a 3-D object must be assembled from a flat piece of material. To find the shape of the flat piece of material for the assembly of the surface, we start with a 3-D model of the surface S and search for an optimal way to flatten it by cutting S. The cutting must start from the boundary and continue toward the interior of S in such a way that (1) the total length of the cutting paths is minimized and (2) if the flat equivalent S' of S in $\mathbb{R}^2$ is glued back along the cuts, the reconstructed surface requires a minimum "stretch" to achieve the original shape of S. The amount of needed stretch, or the deviation of S around a point p from being flat, is measured by the *Gaussian curvature* at p. The Gaussian curvature at a point p of S is the product of the principal curvatures of S at p, i.e., roughly speaking, the product of the maximum and minimum curvatures at the point p among all the curves on the surface passing through p. The larger the absolute value of the Gaussian curvature at p is, the farther S is from being flat at p. Thus, to reduce the stretch, we cut the surface S along points at which the absolute values of the Gaussian curvatures are too large (e.g., with respect to a given threshold value). On the other hand, if the Gaussian curvatures at all points on S are 0, then the

⋆ The research of this author was supported in part by the National Science Foundation under Grant CCF-0515203.

F.P. Preparata, X. Wu, and J. Yin (Eds.): FAW 2008, LNCS 5059, pp. 233–244, 2008.

surface is flat and requires no cuts to flatten it (and no stretch to "reconstruct" the original shape from the flat version).

From the application view point, the surface to be flattened is represented by a triangular mesh $S = (N, M)$, where N is a given set of n points and M is a set of $m = O(n)$ segments connecting the points in N. We assume that the Gaussian curvature at each of the n points is given. (Note that the computation of Gaussian curvature using a discrete representation of the surface is not possible, but it can be approximated, for example, by using a discrete method [9,12].) Each point at which the absolute value of the Gaussian curvature is greater than a given threshold value is required to lie on a cutting path. Clearly, we can use an undirected graph $G = (V, E)$ to represent the mesh surface $S = (N, M)$, by letting $V = N$ and $E = M$. We assume in this paper that such a graph G for S is planar and G admits a planar embedding such that the boundary of S corresponds to the border of the outer face of the planar embedding of G.

Given this setting, the optimal surface flattening problem is defined as follows.

Surface Flattening Problem (SF). *Given a triangular mesh $S = (N, M)$ representing a surface in $\mathbb{R}^3$, $N_A \subseteq N$, which is the set of points with the absolute values of the Gaussian curvatures above a threshold value, and $N_B \subseteq N$, $N_B \neq \emptyset$, which is the set of points on the boundary of the surface S, find a set of cutting paths of the shortest total length along the edges of S that connect every point in N_A to at least one point in N_B.*

Without loss of generality, we assume $N_A \neq \emptyset$ (otherwise, the solution is trivial). Note that in the above definition of the SF problem, the essential requirement is that in the solution, there is a path connecting each point in N_A to some point in N_B. The paths for two different points in N_A can cross or (partially) overlap. Therefore, the union of the paths in a solution should form a certain subgraph in the graph G defined by (N, M), such that the total sum of the edge lengths in this subgraph is minimized.

Since surface flattening is a very important problem in a number of industrial applications, the literature on surface flattening is vast. However, to the best of our knowledge, all previous algorithms for this problem are heuristics without any quality guarantee of their solutions, and no thorough analysis of the computational complexity of the problem has been given. In most cases, the known algorithms seek to find a solution based on an application-specific metric. The various approaches for solving the problem include greedy algorithms that search for seemingly best possible insertions of cuts or darts; for example, Parida and Mudur [11] found cuts during successive flattenings of the mesh triangles, and Aono *et al.* [1,2] inserted darts into a flat cloth to adjust the shape to the surface. Other methods include optimization of various energy functions [8,10,13] and, more recently, algorithms searching for individual shortest cuts passing through points with high absolute values of Gaussian curvatures [12,14]. Sheffer's algorithm [12] finds shortest paths that connect nodes with high absolute values of Gaussian curvatures with the nodes on the boundary of the surface using a minimal spanning tree. Unfortunately, this method does not work well for surfaces

with widely distributed curvatures. Wang *et al.* [14] avoided this problem by first computing a boundary geodesic map and then repeatedly finding a shortest path from a selected node to the surface boundary using this map. Neither of these algorithms guarantees an optimal quality solution. Azariadis *et al.* [3], recognizing the large number of optimization criteria and methods for surface flattening, considered the problem of quality control from the view point of applications. They evaluated many existing surface flattening methods by using "intuitively-acceptable" visualization techniques (they also gave a brief overview of the many surface flattening algorithms). To the best of our knowledge, there are no previous results providing theoretical analysis of the computational complexity of the problem or quality guarantee of the solutions.

In this paper, we prove that the optimal surface flattening problem as specified in [12,14], i.e., the problem of computing cutting paths of the minimum total length along the mesh edges that pass through points with high absolute values of Gaussian curvatures, is NP-hard. This implies that finding an optimal solution for the problem in deterministic polynomial time is unlikely unless P = NP. Furthermore, we show that the surface flattening problem admits a PTAS (polynomial-time approximation scheme), that is, it can be reduced to the problem of computing an optimal Steiner tree on a planar graph, and, as such, can be solved by a $(1+\epsilon)$-approximation algorithm due to Borradaile, Klein, and Mathieu [5] in $O(n \log n)$ time, where $\epsilon > 0$ is any constant. This appears to be a theoretically "best possible" solution for the problem unless P = NP.

2 The NP-Hardness of the Optimal Surface Flattening Problem

In order to show that the optimal SF problem is NP-hard, we first prove the NP-completeness of the following decision version of the problem.

Surface Flattening Decision Problem (SF-D). *Given a triangular mesh $S = (N, M)$ representing a surface in $\mathbb{R}^3$, $N_A \subseteq N$ which is a set of points on S with the absolute values of the Gaussian curvatures above a threshold value, $N_B \subseteq N$ which is a set of points on the boundary of S, and an integer k, does there exist a set of cutting paths along the edges of S that connect every point in N_A to at least one point in N_B whose total length is less than or equal to k?*

To show that the SF-D problem is NP-complete, we must show that (1) the problem is in NP, i.e., it can be solved in polynomial time by a nondeterministic Turing machine (or a given solution can be verified in polynomial time), and (2) the problem is NP-hard (a known NP-complete problem can be transformed in polynomial time to the SF-D problem). Since verifying the first condition is easy, we omit it here. The second part will involve a two-step reduction from a well-known NP-complete problem, the Rectilinear Steiner Tree Problem [6].

Given a set A of points in the plane, a *rectilinear Steiner tree* (RST) for A is a tree connecting all points in A using line segments that are either horizontal or vertical. As opposed to a *spanning tree*, the nodes of an RST (i.e., the endpoints

of the segments of the RST) may include some *Steiner points*, that is, points in the plane that do not belong to A but are needed for the desired connection of the RST. A minimum RST is a tree whose total length of line segments is minimized. The RST problem, stated as follows, has been proved to be NP-complete [6].

Rectilinear Steiner Tree Problem (RSTP). *Given a finite set A of points in the plane and an integer $k > 0$, does there exist a rectilinear Steiner tree for A with a total length no bigger than k?*

In fact, even the special case of the RST problem in which the coordinates of the points in A are all integers was shown to be NP-complete in [6]. In the rest of this paper, whenever we refer to any variations of the (geometric) RST problem, we assume that the coordinates of the points in A are all integers.

For our proof, we start with transforming the RSTP to a slightly different problem, called the *boxed* rectilinear Steiner tree problem (or the boxed RSTP), and proving that the boxed RSTP is also NP-complete. In the boxed RSTP, we choose a certain "box" B_A containing all the points in the set A (as shown below), and require that the sought rectilinear Steiner tree connect all the points of A as well as *at least one point* on the box B_A. Note that for the sought rectilinear Steiner tree to connect all the points in A and at least one point of the box B_A, it is sufficient to consider only a finite set of points on the box B_A, denoted by B. The box B_A and the point set B on B_A are constructed as follows. Create a rectilinear grid R_A using the vertical and horizontal lines on the plane passing through all the points in A. Note that the coordinates of all vertices of R_A are integers. Let $d = 1 + \max\{d_h, d_v\}$, where d_h is the distance between the leftmost and the rightmost vertical lines of the grid R_A, $x = x_l$ and $x = x_r$, and d_v is the distance between the top and the bottom horizontal lines of R_A, $y = y_t$ and $y = y_b$. Clearly, d is an integer value. Further, note that the rectilinear (or L_1) distance between any two points in A is strictly less than $2d$. The box B_A is defined by the two vertical lines $x = x_1 = x_l - 2d$ and $x = x_2 = x_r + 2d$ and the two horizontal lines $y = y_1 = y_b - 2d$ and $y = y_2 = y_t + 2d$. That is, B_A consists of four line segments $((x_1, y_1), (x_2, y_1))$, $((x_2, y_1), (x_2, y_2))$, $((x_1, y_2), (x_2, y_2))$, and $((x_1, y_1), (x_1, y_2))$. The points in the set B are those on the intersection of the grid R_A and the four segments of B_A. See Fig. 1 for an example. Let R_B denote the finite portion of the grid R_A contained inside the box B_A, together with the box B_A as its boundary. Clearly, the coordinates of all vertices of R_B are also integers; furthermore, the lengths of edges of R_B are all integers as well.

Boxed Rectilinear Steiner Tree Problem (BRSTP). *Given a finite set A of points in the plane and an integer $k > 0$, and suppose the box B_A containing A and the point set B on B_A have been determined as above, does there exist a rectilinear Steiner tree for A that also includes at least one point in B with a total length less than or equal to k?*

Note that we can restrict a rectilinear Steiner tree T, as a solution to RSTP or BRSTP, to the grid R_A or R_B in the sense that any Steiner point of the tree T not on a grid vertex can be moved to a grid vertex without changing the

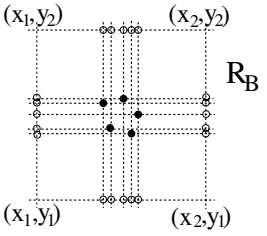

Fig. 1. The points in A are marked as solid dots, and the points in B are marked as empty dots

total length of the resulting rectilinear Steiner tree T'. We have the following statement on A and the grid R_B whose vertices all have integer coordinates only.

Lemma 1. *The boxed rectilinear Steiner tree problem (BRSTP) is NP-complete.*

Proof. Our transformation is from the rectilinear Steiner tree problem (RSTP) (with the points in A all having integer coordinates) to the boxed rectilinear Steiner tree problem (BRSTP). The ideas of the proof are not complicated but the details are a little tedious. Due to the space limit, the detailed proof is omitted here and can be found in the full version of the paper. □

We now show the NP-completeness of the surface flattening decision problem (SF-D) by using a transformation from the NP-complete boxed rectilinear Steiner tree problem on R_B whose vertices all have integer coordinates only. The instance of the SF-D problem produced by the transformation, however, may have nodes with non-integer coordinates.

Theorem 2. *The surface flattening decision problem (SF-D) is NP-complete.*

Proof. Our idea for the proof, based on a transformation from the integer version of BRSTP to SF-D, is fairly straightforward, yet the details of the transformation are a little tedious.

Using the grid R_B on the plane (i.e., the box B_A together with the portion of R_A enclosed in and bounded by B_A), we construct, in polynomial time, a triangular mesh surface $S = (N, M)$ in $\mathbb{R}^3$ such that the solutions to BRSTP on R_B and to SF-D on S are equivalent.

To define S in $\mathbb{R}^3$, we need to specify the x-, y-, and z-coordinates of all the points in N as well as the set M of segments (including their lengths) connecting the points of N. To begin, we include in S all vertices and segments of R_B, initially setting the z-coordinates of all these vertices to 0. This creates a rectangular mesh on the xy-plane (with $z = 0$). To obtain from this planar rectangular mesh a triangular mesh in $\mathbb{R}^3$, we add to the rectangular mesh more vertices and edges: Each of the newly added vertices lies at the intersection of the two diagonals of every cell of the rectangular mesh, and the new edges are the "diagonal" segments connecting each of these diagonal intersection points with its four cell vertices. We denote the set of points thus added by N_D ($N_D \subset N$). Also, we let

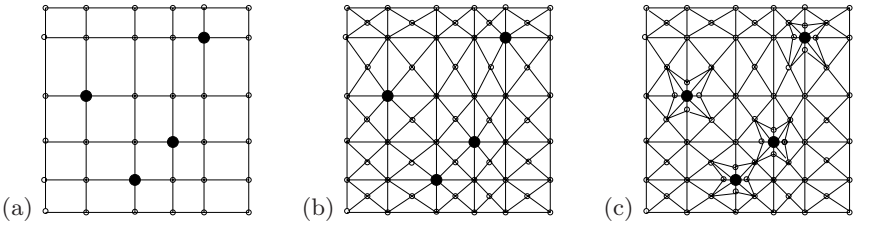

Fig. 2. Illustrating the steps for constructing the mesh surface S (S is vertically projected onto the xy-plane in these figures). (a) $S = R_B$ (the points of N_A are marked by solid dots). (b) The addition of points in N_D lying at the intersections of the diagonals of all the cells of R_B and the addition of the diagonal segments. (c) The addition of points in N_K and the addition of segments between the points in N_D and the neighboring points in N_K.

N_A and N_B denote the subsets of points in N that correspond to the points of A and B on R_B, respectively. Figure 2(a) shows the initial rectangular mesh of S ($= R_B$) in which the points of N_A are marked by large solid dots; Figure 2(b) shows the modified mesh with the points of N_D and the "diagonal" segments.

The next step is to elevate (i.e., increase the z-coordinates of) all points of N_A and N_D in $\mathbb{R}^3$. The necessity of elevating the points of N_A is obvious: To ensure their absolute Gaussian curvatures to be sufficiently large. We elevate the points of N_D to ensure that no points of N_D belong to any shortest path along the edges of S between any two points of S that correspond to two arbitrary vertices of the initial grid R_B (so that any "good" cutting path on S will follow only the edges of S that correspond to those of R_B). Without loss of generality, we assume that the length of the shortest edge of the original grid R_B is one unit. Let l be the length of the longest edge of R_B, and c be the number of edges of R_B. Then we set the z-coordinates of all points in N_D to be $2l + 1$ and the z-coordinates of all points in N_A to be $1/(9c)$.

Let g_0 be the absolute value of the Gaussian curvature at the intersection point p of the diagonals of a 1×1 square such that the z-coordinate of p is $2l + 1$ and all the vertices of the square are on the xy-plane. Since the length of the shortest edge of R_B is 1, the absolute value of the Gaussian curvature at any point of N_D is no bigger than g_0. We set the threshold curvature value for the SF-D problem on S to be g_0. To ensure a high absolute value of the Gaussian curvature at each point $p_a \in N_A$ on S, we build a "steep" structure around p_a on S, as follows. We create four points around every point $p_a \in N_A$, and denote the set of all the points thus created by N_K. Specifically, for each $p_a \in N_A$, we place points $p_i^a \in N_K$, $i = 1, 2, 3, 4$, on each of the four segments of the original rectangular mesh R_B adjacent to p_a, such that the z-coordinate of each p_i^a is 0 and p_i^a is at a distance d_0 from the vertical projection point of p_a on the xy-plane. The fixed value d_0 is chosen to be sufficiently small to satisfy the following two conditions: (1) $d_0 \leq 1/(9c)$, and (2) the absolute value of the Gaussian curvature at any $p_a \in N_A$ is larger than the threshold value g_0. (The value d_0 can be calculated from the fixed values g_0, l, and c.) The points in

N_K, all on the xy-plane, are added to the set N to make their corresponding points $p_a \in N_A$ locally "steep" (thus attaining high absolute values of Gaussian curvatures at p_a), and they split each edge of R_B adjacent to every $p_a \in N_A$ into two segments in M. Lastly, we add to M the segments connecting each point $p_d \in N_D$ with any point $p_i^a \in N_K$ lying on the boundary of the cell of R_B containing p_d.

This completes the construction of S. See Fig. 2(c) for an example of the resulting mesh surface S (as projected vertically onto the xy-plane). Clearly, the construction of S takes polynomial time since S consists of $O(|A|^2)$ vertices and $O(|A|^2)$ segments.

It remains to show that BRSTP has a solution T_{BP} of a total length less than or equal to k on R_B if and only if SF-D has a solution C_{SF} of a total length less than or equal to $k + 1/2$ on S, for any given integer $k > 0$.

Before we show the equivalence of these two solutions, recall that the reason for adding the points of N_D to N is to create a triangular mesh for the surface flattening problem. However, we need to prevent any "good" cutting paths of S from going through any points of N_D so that the cutting paths on S follow only those segments of S corresponding to the edges of R_B (as required by a BRSTP solution on R_B). By elevating the points of N_D high enough to make their adjacent segments longer than any of the edges of R_B and by making the heights of all the points of N_A very small, we make sure that no points of N_D belong to any shortest path along the segments of S connecting any two points in $N - N_D$. This is because a path between any two points in $N - N_D$ through a "diagonal peak point" $p_d \in N_D$ of any cell of R_B is longer than a path along the border of that cell. Thus, "good" cutting paths connecting points in $N - N_D$ and along the segments of S use only segments that correspond to the edges of R_B. Without loss of generality, we assume that any solution C_{SF} for the SF-D problem on S, that is, the union of a set of cutting paths for N_A on S, uses only segments of S that correspond to the edges of R_B (otherwise, we can always replace any cutting path in C_{SF} passing through a point in N_D by another cutting path using only the segments along the borders of the cells of R_B without increasing the total length of the resulting SF-D solution).

Now, suppose T_{BP} is a solution to BRSTP on R_B of a total length less than or equal to k. Note that since the lengths of all edges of R_B are integers, the total length $|T_{BP}|$ of T_{BP} must be an integer as well. Let C_{SF} consist of all the segments of S whose vertical projections onto the xy-plane lie entirely on the edges of T_{BP}. Then C_{SF} is a feasible solution to SF-D since every point in N_A is connected to a point in N_B along the segments of $S \cap C_{SF}$ (this follows from the fact that T_{BP} is an RST for A and connects to at least one point of B). Note that the segments of C_{SF} correspond to the edges of T_{BP} on R_B, plus connections to a little "tip" at each point $p_a \in N_A$. Every such little tip at $p_a \in N_A$ adds a total length less than $4 \times 1/(9c) < 1/(2c)$ to the total length of T_{BP} for the SF-D solution C_{SF}. Thus, a total length less than $|A| \times 1/(2c) \leq c \times 1/(2c) = 1/2$ due to the "tips" over all the points of N_A is added to the total length of T_{BP} for

the SF-D solution C_{SF}, implying that the total length $|C_{SF}|$ of C_{SF} is no bigger than $|T_{BP}| + 1/2 \leq k + 1/2$.

Conversely, suppose C_{SF} is a solution to SF-D on S of a total length less than or equal to $k + 1/2$. Let T_{BP} be the set of edges on R_B corresponding to the segments of C_{SF}. Since, unlike C_{SF}, T_{BP} does not contain the "tips" at the points of N_A, its total length $|T_{BP}|$ is strictly smaller than $|C_{SF}| \leq k + 1/2$. Further, since the total length of T_{BP} must be an integer, the largest possible integer value for $|T_{BP}|$ that is less than $k + 1/2$ is k. Besides, as argued in the previous paragraph, the total contribution of all the "tips" to the total length of C_{SF} is strictly less that $1/2$. Thus, $|T_{BP}| \leq k$ holds. If T_{BP} forms a tree, then we are done, since it contains all the points of A and at least one point of B. However, note that C_{SF} is the union of cutting paths along the segments of S corresponding to the edges of R_B, and such a cutting solution, although connecting each point of N_A to some point of N_B, need not form a tree structure. That is, T_{BP} may not be a desired RST for BRSTP on R_B. Actually, T_{BP} may consist of multiple connected components, and each such component may contain some cycles. Hence, our remaining task is to convert T_{BP} to an RST for A on R_B that touches at least one point in B, *without increasing* the total length of the resulting T_{BP}.

Observe that since C_{SF} is a feasible cutting solution to SF-D on S, every connected component of T_{BP} must contain at least one point of B and some points of A (those components containing no points of A can be safely removed from any further consideration). We first remove from T_{BP} all edges that lie entirely on the boundary of the box B_A (this removal does not yet remove the end vertices of those edges). If any point of $B \cap T_{BP}$ becomes an isolated vertex of T_{BP} after this edge-removal, then remove that point from T_{BP} as well. Clearly, this removal process does not affect the feasibility of the corresponding cutting solution to SF-D on S (i.e., if the corresponding segments and endpoints are removed from C_{SF}), and can only decrease the total length of T_{BP}. But, it can create more connected components of T_{BP}, each of which still contains at least one point of B (but, each such point of B is now of degree exactly one in T_{BP}). Note that for any two points of A, the rectilinear (or L_1) distance between them is strictly less than $2d$. We "merge" any two distinct connected components C_1 and C_2 of T_{BP} into one component, as follows: Remove from T_{BP} all the points of B and their adjacent edges that are contained in C_1 (this decreases the total length of T_{BP} by at least $2d$), and connect an arbitrary remaining vertex of C_1 with an arbitrary vertex of C_2 by a shortest path lying on R_B (this increases the total length of T_{BP} by less than $2d$). The resulting T_{BP}, with at least one less connected component, has a smaller total length than its previous version. Further, T_{BP} still contains all the points of A, and each of its connected components contains at least one point of B. We continue this merge process until T_{BP} has exactly one connected component. At this point, T_{BP} is a connected subgraph on R_B that contains all the points of A and at least one point of B. But, T_{BP} may not yet be a tree. Thus, we remove from T_{BP} any edge (but not its end vertices) without disconnecting T_{BP} (this can only further

decrease the total length of T_{BP}). We continue this edge-removal process until T_{BP} becomes a tree, which is a sought RST for A touching at least one point of B on R_B, and its total length is no bigger than k. Hence, the resulting T_{BP} is a solution to BRSTP for A on R_B with $|T_{BP}| \leq k$.

Finally, it is easy to see that given a solution C_{SF} to SF-D on S, a corresponding solution T_{BP} to BRSTP on R_B can be obtained in polynomial time in terms of the input size $|A|$. □

Because the decision version of the surface flattening problem is NP-complete, the optimization version of this problem is NP-hard.

3 A $(1 + \epsilon)$-Approximation SF Algorithm

Since the optimal surface flattening problem (SF) is NP-hard, it is unlikely that a deterministic polynomial time algorithm for optimally solving this problem is possible unless P = NP. Thus, we present an efficient method for finding a provably good approximate solution for the problem.

Our approach is to solve the SF problem using a graph representation. We model the mesh surface $S = (N, M)$ using an undirected weighted graph $G = (V, E)$, where $V = N$ and $E = M$, i.e., the vertices of G correspond to the point set of S and the edges of G correspond to the line segments of S. We let $V_A \subseteq V$ be the subset of vertices corresponding to the points of N_A whose absolute values of the Gaussian curvatures are greater than the threshold value. The weight of an edge $e = (v, w)$ in E is equal to the length of the segment between the mesh points represented by vertices v and w. Since S is a bounded surface in $\mathbb{R}^3$, the graph G is planar and a planar embedding of G can be easily specified by S such that the boundary of S corresponds to the border of the outer face of the planar embedding of G. Clearly, the one-to-one correspondence between the mesh points on the boundary of S and the vertices on the outer face of G defines the vertex subset $V_B \subseteq V$ as corresponding to $N_B \subseteq N$.

Given the graph representation G of the surface S, observe that the union of the optimal cutting paths connecting some boundary points of S with all the points of N_A along the segments of S corresponds to a set of trees, or a forest, F, in G whose roots all lie on the outer face. It is required that the total length of these optimal cutting paths (or trees) be minimized. The reason that the union of the optimal cutting paths for S must form a forest is as follows. Suppose the union, F, in G contains a cycle. Then we can remove from F any edge (but not its end vertices) on the cycle, still retaining a feasible cutting solution for S; but, the solution thus obtained has a smaller total length than that of F, a contradiction to the assumption that F corresponds to the union of the optimal cutting paths for S.

The optimal surface flattening problem defined on the graph model G is as follows.

Terminal Cutting Problem on a Planar Graph (TCPG). *Given an embedded undirected weighted planar graph $G = (V, E)$ representing a mesh surface*

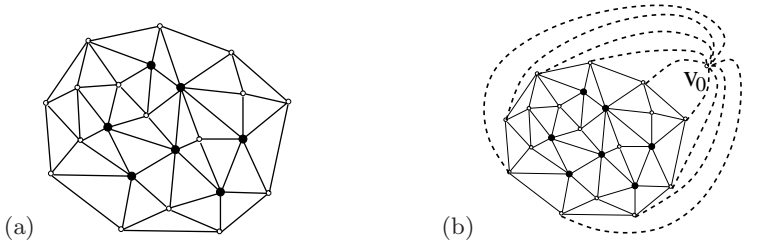

Fig. 3. (a) A planar graph G representing the surface S (the vertices of V_A are denoted by solid dots, and the vertices of V_B are on the outer face of G). (b) The planar graph G' augmented by adding the supernode v_0 and the edges connecting v_0 with all the vertices of V_B.

S in $\mathbb{R}^3$, $V_A \subseteq V$, which is a set of terminals (corresponding to the points of A on S whose absolute values of the Gaussian curvatures are above a threshold value), and $V_B \subseteq V$, which is the set of vertices on the outer face of G (corresponding to the boundary nodes of S), find a subgraph F of G with the minimum total sum of edge weights connecting every vertex of V_A to at least one vertex of V_B.

We solve the TCPG problem by transforming it to the optimal Steiner tree problem on planar graphs, which is known to be NP-hard [6], and applying an $O(n \log n)$ time $(1 + \epsilon)$-approximation algorithm for the optimal Steiner tree problem on planar graphs [5] to finish the job.

Optimal Steiner Tree Problem on a Planar Graph (OSTPG). *Given an undirected planar graph $H = (V_H, E_H)$ with nonnegative edge weights and a set $T \subseteq V_H$ of terminals, find a tree F' in G with the minimum total sum of edge weights that contains all terminals of T.*

The key to the transformation from TCPG to OSTPG is to make certain changes to the graph G while preserving the planarity of the resulting graph G' and to specify which of the vertices in G' are to be terminals in T for OSTPG. Clearly, T should include all the vertices of V_A. But, since TCPG requires that each vertex of V_A be connected with some vertex on the outer face of G, a feasible solution to OSTPG needed by TCPG must include connections to some vertices of V_B. To satisfy this requirement, we add to G a new vertex v_0, which is also a terminal in T and is called a *supernode*, and add edges that connect v_0 with all the vertices of V_B such that the weights of these added edges are all zero. To preserve the planarity of the resulting embedded graph, v_0 and its edges are all placed in the interior of the outer face of G. Let G' be the new graph thus obtained from G. Figure 3 illustrates the transformation from G to G'. Clearly, G' is planar and has a set of terminals that includes v_0. Furthermore, G' has $O(|V|)$ vertices and $O(|E|) = O(|V|)$ edges. The next lemma shows the equivalence of a solution to TCPG on G and a corresponding solution to OSTPG on G'.

Before we proceed to the next lemma, observe that a non-optimal solution to TCPG is a subgraph of G that may consist of one or more connected components,

each of which contains some vertices of V_A and at least one vertex of V_B and may even contain some cycles (i.e., it may not be a tree). However, when this case occurs, we can, easily in linear time (say, based on depth-first search), remove any edges (but not their end vertices) from each non-tree component without disconnecting it until the component becomes a tree. This edge-removal process does not affect the feasibility of the resulting TCPG solution for G, and the total sum of edge weights of the resulting TCPG solution can only decrease. Thus, without loss of generality, we assume for the rest of this paper that any solution to TCPG consists of $l \geq 1$ connected components in G, each of which is a tree.

Lemma 3. *Let $G = (V, E)$ be an undirected, weighted embedded planar graph representing a mesh surface S, $V_A \subseteq V$ be the subset of vertices representing the set A of points on S whose absolute Gaussian curvatures are above a threshold value, and $V_B = \{v_{b_1}, \ldots, v_{b_k}\} \subseteq V$ be the subset of vertices on the outer face of G representing the nodes on the boundary of S. Furthermore, let $G' = (V \cup \{v_0\}, E \cup \{(v_0, v_{b_1}), \ldots, (v_0, v_{b_k})\})$, and $T = V_A \cup \{v_0\}$ be a set of terminals in G'. Then a solution to TCPG on G can be obtained from a solution to OSTPG on G' (with the same sum of edge weights), and* vice versa. *Moreover, the transformation takes $O(n)$ time, where $n = |V|$.*

Proof. The proof of the equivalence of the solutions to OSTPG and TCPG is not difficult to show. Due to the space limit, the detailed proof is omitted here and can be found in the full version of the paper. □

Using the algorithm for OSTPG by Borradaile, Klein, and Mathieu [5], we obtain a $(1+\epsilon)$-approximate solution to TCPG (and thus to SF) in $O(2^{\text{poly}(1/\epsilon)} n \log n)$ time, for any constant $\epsilon > 0$. Note that for any constant $\epsilon > 0$, $2^{\text{poly}(1/\epsilon)}$ is a (possibly large) constant as well. Hence the running time of our $(1+\epsilon)$-approximation SF algorithm is $O(n \log n)$.

Theorem 4. *The surface flattening problem can be solved by a $(1+\epsilon)$- approximation algorithm in $O(2^{\text{poly}(1/\epsilon)} n \log n)$ time, for any constant $\epsilon > 0$.*

Proof. This follows immediately from Lemma 3 and the PTAS result in [5]. □

For any constant $\epsilon > 0$, we have obtained in $O(n \log n)$ time an approximate solution (i.e., a cutting of the surface S) whose total length is no more than $1+\epsilon$ times the total length of the optimal solution. Due to its NP-hardness, this appears to be a theoretically "best possible" approximation solution for the SF problem.

References

1. Aono, M., Breen, D.E., Wozny, M.J.: Modeling methods for the design of 3D broadcloth composite parts. Computer-Aided Design 33(13), 989–1007 (2001)
2. Aona, M., Denti, P., Breen, D.E., Wozny, M.J.: Fitting a woven cloth model to a curved surface: Dart insertion. IEEE Computer Graphics and Applications 16(5), 60–70 (1996)

3. Azariadis, P.N., Sapidis, N.S.: Planar development of free-form surfaces: Quality evaluation and visual inspection. Computing 72(1-2), 13–27 (2004)
4. Borradaile, G., Kenyon-Mathieu, C., Klein, P.N.: A polynomial-time approximation scheme for Steiner tree in planar graphs. In: Proc. of 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1285–1294 (2007)
5. Borradaile, G., Klein, P.N., Mathieu, C.: Steiner tree in planar graphs: An $O(n \log n)$ approximation scheme with singly exponential dependence on epsilon. In: Proc. of 10th International Workshop on Algorithms and Data Structures, pp. 276–287 (2007)
6. Garey, M.R., Johnson, D.S.: The rectilinear Steiner tree problem is NP-complete. SIAM Journal on Applied Mathematics 32(4), 826–834 (1977)
7. Karp, R.: On the computational complexity of combinatorial problems. Networks 5, 45–68 (1975)
8. Kim, S.M., Kang, T.J.: Garment pattern generation from body scan data. Computer-Aided Design 35(7), 611–618 (2003)
9. Kobbelt, L.P., Bischoff, S., Botsch, M., Kähler, K., Rössl, C., Schneider, R., Vorsatz, J.: Geometric modeling based on polygonal meshes. In: Eurographics 2000 Tutorial (2000)
10. McCartney, J., Hinds, B.K., Seow, B.L.: The flattening of triangulated surfaces incorporating darts and gussets. Computer-Aided Design 31(4), 249–260 (1999)
11. Parida, L., Mudur, S.P.: Constraint-satisfying planar development of complex surfaces. Computer-Aided Design 25(4), 225–232 (1993)
12. Sheffer, A.: Spanning tree seams for reducing parameterization distortion of triangulated surface. In: Proc. of International Conference on Shape Modeling and Applications, pp. 61–68 (2002)
13. Wang, C.L., Smith, S.F., Yuen, M.F.: Surface flattening based on energy model. Computer-Aided Design 34(11), 823–833 (2002)
14. Wang, C.L., Wang, Y., Tang, K., Yuen, M.F.: Reduce the stretch in surface flattening by finding cutting paths to the surface boundary. Computer-Aided Design 36(8), 665–677 (2004)