High Quality Compatible Triangulations

Vitaly Surazhsky   Craig Gotsman

Center for Graphics and Geometric Computing
Dept. of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel
{vitus|gotsman}@cs.technion.ac.il

ABSTRACT

Compatible meshes are isomorphic meshing of the interiors of two polygons having a correspondence between their vertices. Compatible meshing may be used for constructing sweeps, suitable for finite element analysis, between two base polygons. They may also be used for meshing a given sequence of polygons forming a sweep. We present a method to compute compatible triangulations of planar polygons with a very small number of Steiner (interior) vertices. Being close to optimal in terms of the number of Steiner vertices, these compatible triangulations are usually not of high quality, i.e., do not have well-shaped triangles. We show how to increase the quality of these triangulations by adding Steiner vertices in a compatible manner, using several novel techniques for remeshing and mesh smoothing. The total scheme results in high-quality compatible meshes with a small number of triangles. These meshes may then be morphed to obtain the intermediate triangulated sections of a sweep, if needed.

Keywords: mesh generation, compatible triangulations, remeshing

1. INTRODUCTION

In CAE, swept volumes, sometimes called two and one half dimensional volumes, are frequently constructed between two base polygons given with a correspondence between their vertices. To discretize a swept volume for finite element analysis, it is necessary to mesh the interiors of the sequence of polygonal cross-sections forming the sweep, usually introducing interior (Steiner) vertices, in a manner such that the mesh is isomorphic, valid and well-formed within all the polygons. This mesh is said to be *compatible* with all the polygons. See Figure 1 for an example. The result is a set of prisms defining the sweep, whose edges are the so-called "ribs" of the sweep [15].

In the case where only the two base polygons of the sweep are given, it is possible to automatically generate the intermediate polygons by a process known as *morphing*. The morphing problem, in general, is to smoothly transform one given polygon, the *source*, into another, the *target*, over time. Constructing the sweep volume may be considered a morphing problem by thinking of the sweep axis as the time axis of the morph. Morphing has been the subject of much research over recent years and has wide practical use in areas such as computer graphics, animation and modeling.

The naive approach to the morphing problem is to decide that the polygon vertex trajectories are straight lines, where every feature of the shape travels with a constant velocity along the line towards the corresponding feature of the target during the morph. However, this simple approach

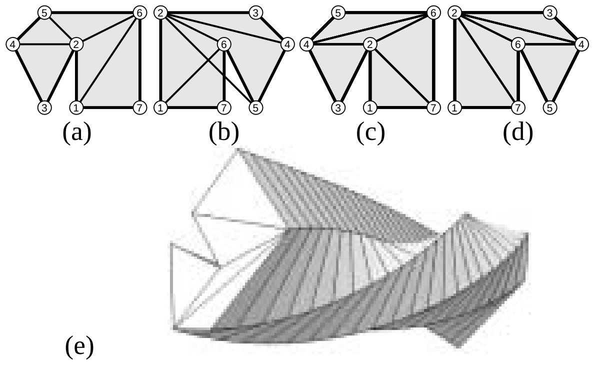


Figure 1: The concept of compatible triangulations of corresponding polygons. Vertex correspondence is denoted by digits. (a),(b) Non-compatible triangulation. (c),(d) Compatible triangulation. (e) The sweep with bases (c) and (d).

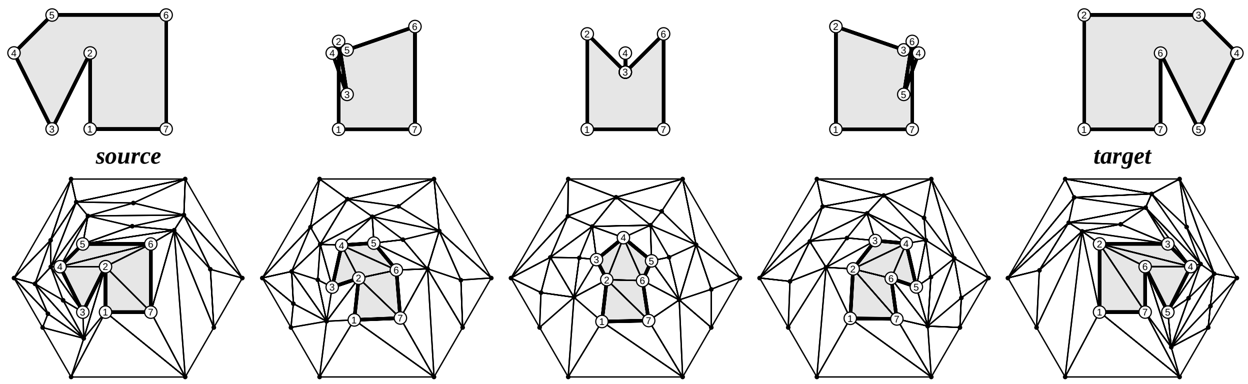


Figure 2: Morphing of two corresponding polygons. The correspondence is denoted by digits. Leftmost polygon is source and rightmost is target. Note that the correspondence implies some rotation during the morph. *Top row:* The linear morph resulting in self-intersecting intermediate polygons. *Bottom row:* The morph of the polygons generated by embedding the source and target into compatible triangulations and applying the method of [8] or [16] guarantees that the intermediate polygons are also simple.

can lead to undesirable results. The intermediate shapes can vanish, i.e. degenerate into a single point, or self-intersect even though the source and target are simple. Even if the linear morph is free of self-intersections and degeneracies, its intermediate shapes may have areas or distances between features far from those of the source and target, resulting in a "misbehaved" morph. See the top row of Figure 2. Most of the research on solving the trajectory problem for morphing concentrates on trying to eliminate self-intersections and preserving the geometrical properties of the intermediate shapes. Numerous existing methods achieve good results for many inputs, (e.g. [13] and [14]), yet, only the methods that use compatible triangulations are able to *guarantee* any properties of the resulting morph.

In order to perform finite-element analysis on a swept volume—a sequence of corresponding simple polygonal cross sections—it is necessary to mesh the polygon interiors in a compatible manner. In this work we concentrate on compatible *triangulations.* Compatible meshing is not always possible unless Steiner vertices are introduced into the interior of the polygons. The main challenge is then to minimize the number of Steiner vertices to the least needed to achieve compatibility. Unfortunately, this can be as much as $\Omega(n^2)$, where n is the number of vertices of the polygons. In the first work on this problem, Aronov et al. [2] provided two constructions which result in quite a large number of Steiner vertices. In their work on polygon morphing, Surazhsky and Gotsman [17] improved Aronov et al.'s so-called "spiderweb" method to significantly reduce the number of Steiner vertices required. Kranakis and Urrutia [11] presented a completely different method in which the number of Steiner vertices introduced depends on the number of inflection vertices of the two polygons. Gupta and Wenger [9] described an algorithm which uses minimal-link polylines in the polygon.

While compatible triangulations of polygons with a very small number of Steiner vertices are definitely an advantage from a complexity point-of-view, it appears that these triangulations are naturally not well-formed. They tend to contain long skinny triangles which cannot be adjusted to improve the triangle shape significantly. Hence a major challenge in our application is to introduce as small a number of Steiner vertices as possible, yet obtain two triangulations with decent quality, and maintain compatibility of the triangulations throughout the process. We call this process *compatible remeshing.* This was attempted in the work of Alexa et al. [1], who start off with compatible triangulations of polygons, and introduce Steiner vertices in order to improve the quality of the triangulation. They, however, start from a large number of Steiner vertices, and thereafter increase this number significantly, in order to achieve triangulation of good quality. This results in compatible triangulations that are overly complex.

Our main contribution in this paper is a new method to compatibly triangulate two planar polygons with a very small number of Steiner vertices, and a new remeshing technique, including a novel smoothing component. This introduces a small number of extra Steiner vertices, yet achieves a high-quality triangulation while maintaining the compatibility of the two triangulations.

Meshing for sweep generation has been attempted before by a variety of authors in the meshing community (e.g. [5], [10], [15]). Their basic approach is to generate a mesh for a subset of the cross-section polygons, usually just one of the sweep bases, and then project this mesh somehow onto the other polygons. Beyond the fact that this certainly does not guarantee that the result will be a valid triangulation, there are also no guarantees for the quality of the triangulation even if it were valid. Our solution, taking into account both sweep bases (and theoretically all intermediate polygons), solves all these problems.

If only the base polygons of the sweep are given, the intermediate polygons, with their corresponding compatible triangulations, may be generated using the morphing methods of Surazhsky and Gotsman [17]. This is done by reducing the problem to that of morphing compatible planar *triangulations* with a common convex boundary, in which the polygon is embedded, as described by Floater and Gotsman [8] and Surazhsky and Gotsman [16]. Two corre-

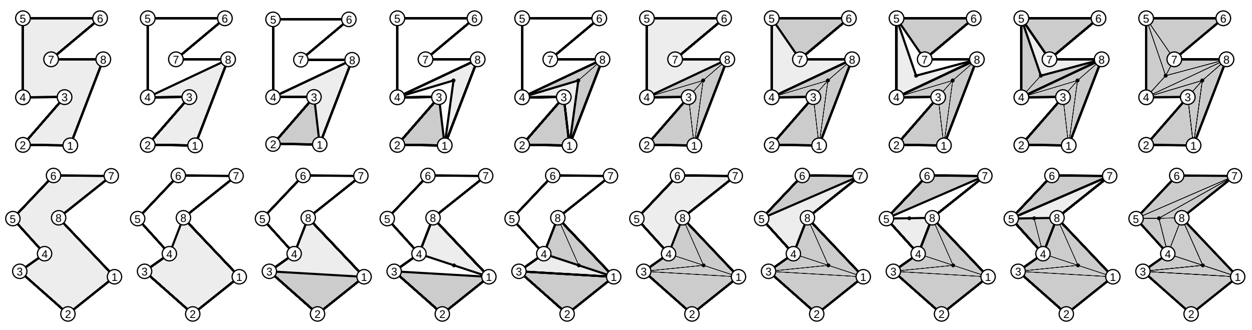


Figure 3: Compatible triangulations of two polygons (one on each row), whose vertex correspondence is denoted by digits. The light grey region denotes the current polygon during the recursion. The grey regions are regions already triangulated. Thick segments are minimal-link polylines, which recursively partition the polygons.

sponding point sets admit a compatible triangulation if there exists a triangulation of one point set which, when induced on the second point set by the correspondence, is a legal triangulation there too. The morphing method of Floater and Gotsman [8] is based on the convex representation of triangulations using barycentric coordinates, first introduced by Tutte [20] for graph drawing purposes, and later generalized by Floater [7] for parameterization of 3D meshes. This avoids many of the problems associated with morphing, and basically guarantees that the triangulation remains valid (i.e. compatible with the source and target) throughout the morph.

To embed the two polygons in a triangulation, first compatibly triangulate the polygon interiors. Then circumscribe the two polygons in a common convex enclosure and compatibly triangulate the two resulting annuli between the polygons and the enclosure (possibly requiring Steiner vertices) [3]. This results in two compatible triangulations with a common convex boundary, in which the polygons are embedded. Morphing these triangulations using the methods of [8] or [16] will then result in a valid (compatible) morph of the two polygons. See the bottom row of Figure 2.

2. NEAR-OPTIMAL COMPATIBLE TRIANGULATIONS

2.1 Previous work

As already stated, Kranakis and Urrutia [11] presented two different methods to compatibly triangulate two polygons in which the number of Steiner vertices introduced depends on the number of the polygons' inflection vertices. The first algorithm produces a rather large number, $O((k+l)^2)$, of Steiner vertices, where k and l are the number of the two polygons' reflex vertices respectively. The second algorithm introduces, at most, $O(kl)$ Steiner vertices, but its drawback is that it may add Steiner vertices on the polygon boundaries, which some applications do not allow. Furthermore, enlarging the boundary might prevent this algorithm from being used as a black-box in a recursive manner, as the algorithm might not terminate.

Gupta and Wenger [9] described an algorithm, seemingly the best so far, which constructs the compatible triangulation based on minimal-link polylines inside the polygons P and Q. A minimal-link polyline is a path of straight-line segments between two vertices lying entirely in the polygon interior, having a minimal number of segments. The idea behind their algorithm is the following: First, compute an arbitrary triangulation T_p of P. Using edges of T_p it is possible to partition P into sub-polygons such that the number of links in minimal-link polylines in those sub-polygons is no more than a small constant (e.g. 5). Then the corresponding partition of Q is constructed using non-intersecting minimum-link polylines. The vertices of these polylines are the Steiner vertices of the triangulations. These corresponding sub-polygons are then compatibly triangulated, usually requiring a relatively small number of Steiner vertices due to the properties of the partition. The resulting compatible triangulations have $O(M\log n + n\log^2 n)$ triangles, where M is the number of triangles in the optimal solution. Theoretically, this is good, except that the constant factor is quite large (approximately 40, according to the authors), so it is not very practical for smaller inputs. Another drawback of this algorithm is that it is not symmetric in P and Q. The choice of the triangulation T_p of P strongly influences the resulting compatible triangulations and the number of Steiner vertices. Thus, overall, in some very simple cases when two triangulations may be compatibly triangulated without requiring Steiner vertices, the algorithm can, nonetheless, introduce a large number of Steiner vertices. From a practical point of view, the algorithm involves implementing many state-of-the-art computational geometry algorithms developed over the last two decades. As a consequence, an implementation of the algorithm is currently not available, and thus, it is impossible to compare this algorithm with other algorithms for finding compatible triangulations.

2.2 Our algorithm

Our algorithm is similar in spirit to that of Gupta and Wenger [9], namely, it is based on the idea of partitioning polygons using minimum-link polylines. However, we believe our algorithm is much simpler. Given two polygons P and Q with a correspondence between their vertices, we find a pair of vertices u and v with a minimal-link polyline between them in one of the polygons, and a corresponding polyline in the second. After the shorter polyline is refined to the same number of vertices as the longer one, the two polylines compatibly partition both of the input polygons into two sub-polygons. The vertices of these polygons are the Steiner vertices. We then apply the algorithm recursively on these two sub-polygons. The process terminates when the input polygons contain only three vertices, namely, the polygons have become triangles.

Note that if it is possible to compatibly triangulate the two polygons *without any* Steiner vertices, our algorithm will do so, as opposed to most of the other algorithms. Since this is the case for many inputs, our algorithm has a significant advantage.

We still need to show how to find a pair of vertices *u* and *v* that minimizes the number of links in the partitioning polylines. To achieve this, we employ the method of Suri [18], who showed how to find the minimum-link path between two *given* vertices in a simple polygon in O(n) time, where n is the number of polygon vertices. In a subsequent work, Suri [19] showed how a simple polygon can be preprocessed in O(n) time in order to query the number of links of the minimum-link path between two given vertices of the polygon in O(log n) time. Thus, we can query all possible vertex pairs of the polygon in O(n^2 log n) time using this algorithm. Hence, in this manner we may determine which pair is best to use, and then employ the first algorithm to actually compute the paths.

Accordingly, in order to find the best path for both polygons we query the two polygons for the minimum-link distance and choose the pair that has the best (minimal) value of the maximum between two distances. Namely, we choose the pair (u,v) which satisfies:

$$(u, v) = \arg\min_{u,v \in P} \max(dist_P(u, v), dist_Q(u, v))$$

In practice, this pair is not unique. Therefore, we choose the pair that will partition the polygons into sub-polygons which are as balanced as possible, in order to reduce the overall algorithm complexity. This can be easily done by comparing the indices of the polygon vertices. More formally, if the polygon vertices are $v_1, \ldots, v_n$ and n is the size of the polygon, we look for:

$$(v_i, v_j) = \arg\max_{v_i, v_j \in P\ j>i} min(j-i, i-j+n)$$

We believe that the time complexity for finding the optimal vertex pair(s) of the polygon(s) can be further improved to O($n$log$n$) by exploiting the existing preprocessed data structure for the queries, instead of using the query procedure for a specific pair as a black box.

We must still show that the algorithm terminates, since when the polygon P is partitioned into two sub-polygons P_1 and P_2, theoretically the size of P_1 or P_2 (or both) can be identical to that of P, and if this repeats, the algorithm can run infinitely. In general, to prevent such cases we should check that the size of the partitioned polygon can be the same as P only once. If the size stays the same after two iterations, the algorithm should backtrack and choose another vertex pair for the partition polylines. This, theoretically, results in the exponential time complexity. However, in practice (we have tested the algorithm over numerous, very complex inputs), even the case when the size of the polygons repeats itself twice does not occur. Thus, although we cannot prove it at this time, we believe that the average total time complexity is less than O(n^3log$n$).

See Figure 3 for an illustration of the various stages of the compatible triangulation algorithm.

3. MESH IMPROVEMENT

Our compatible triangulation algorithm generates a small number of Steiner vertices, at locations which have not necessarily been optimized for mesh quality. It is possible to improve these meshes by smoothing them (moving the vertices), or remeshing them (changing the connectivity). In this section we describe methods for these two operations which we believe are also of independent interest.

3.1 Weighted angle-based smoothing

Zhou and Shimada [21] presented an effective and easy-to-implement angle-based mesh smoothing scheme. They show that the quality of the mesh after angle-based smooth-

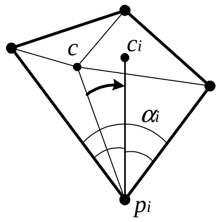
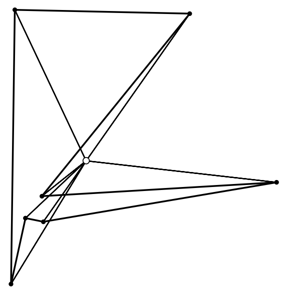
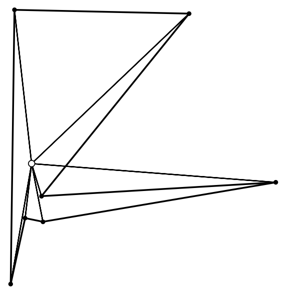


Figure 4: Weighted angle-based smoothing: c_i is obtained by rotation of c around p_i to coincide with the bisector of α_i.


(a)


(b)

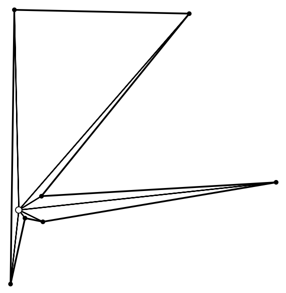

(c)

Figure 5: Comparison of smoothing methods. (a) Laplacian. (b) Angle-based [21]. (c) Weighted angle-based.

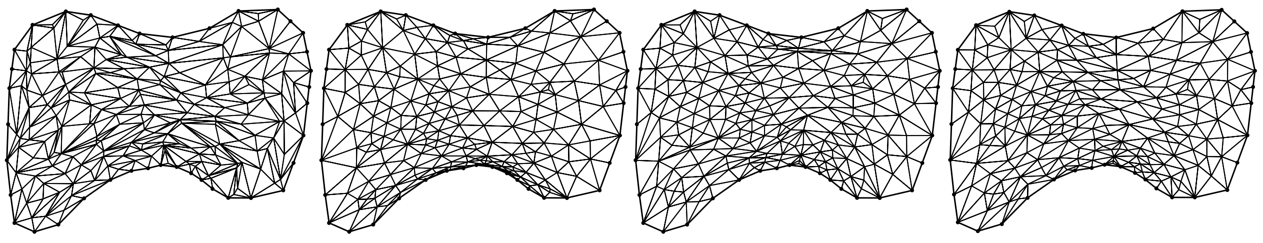


Figure 6: Comparison between smoothing methods. (a) The original mesh. (b) Laplacian. (c) Angle-based [21]. (d) Weighted angle-based. See Table 1 for a quantitative comparison.

ing is much better than after Laplacian smoothing. Moreover, the chance that the scheme will produce inverted (invalid) faces is much less than that in Laplacian smoothing. Unfortunately, this is true mostly for meshes whose vertices have degrees close to the average degree, namely, the mesh connectivity is very regular. When the mesh has more irregular connectivity, the scheme may fail. In applications involving meshes with very distorted (long and skinny) triangles, a more robust smoothing scheme is critical. We propose a very simple improvement to the original angle-smoothing scheme, which significantly reduces the chances of inverted triangles and improves the quality of the resulting mesh. Furthermore, it has almost the same computational cost per iteration and a lower total computational cost due to better convergence.

The original scheme attempts to make each pair of adjacent angles equal. Given a vertex c and its neighbors $p_1 ... p_k$, where k is the vertex degree, we want to move c in order to improve the angles of the triangles incident on c. Let α_i be the angle adjacent to p_i in the polygon $p_1 ... p_k$. We define c_i to be the point lying on the bisector of α_i such that $\|(p_i, c_i)\| = \|(p_i, c)\|$, namely, the edge (p_i, c) is rotated around p_i to coincide with the bisector of α_i. See Figure 4. The new position of c is defined as the average of all c_i for all neighbors. That is:

$$c_{new} = \frac{1}{k}\sum_{i=1}^{k} c_i. \qquad (1)$$

We improve this scheme by introducing weights into (1). For a small angle α_i it is difficult to guarantee that c_{new} will be placed relatively close to the bisector of α_i. Since α_i is itself small, the large deviation of c_{new} from the bisector of α_i will create angles much smaller than $\alpha_i/2$. Thus, the resulting mesh will be of a poor quality. To prevent this, we modify (1) in the following way:

$$c_{new} = \frac{1}{\sum_{i=1}^{k} 1/\alpha_i^2} \cdot \sum_{i=1}^{k} \frac{1}{\alpha_i^2} \cdot c_i. \qquad (2)$$

Namely, the c_i for small angles α_i will carry more weight than for large angles. To demonstrate the robustness of our improvement, see Figure 5, Figure 6 and Table 1.

Despite the superior results of our weighted angle-based scheme, it still cannot guarantee that the new vertex position forms a valid triangulation. Similarly, the convergence of our scheme as well as the original scheme cannot be guaranteed in cases when the given mesh has invalid (in-

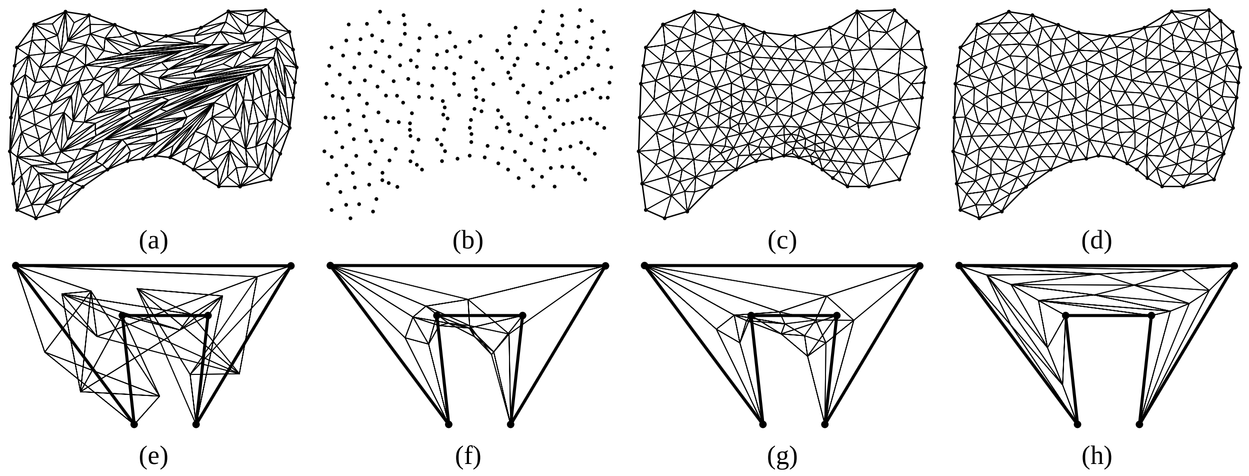


Figure 7: Area-based remeshing. (a) Triangle areas of the mesh from Figure 6 are equalized. (b) Discarding the edges of (a) reveals a uniform vertex distribution. (c) Mesh obtained by alternation of angle-based smoothing and weighted angle-improving edge-flips. (d) Mesh obtained by alternation of area equalization and edge-flips. (e) A polygon with random distribution of interior vertices. (f) Laplacian smoothing of (e). (g) Angle-based smoothing of (e). (h) Area equalization of (e).

verted) triangles or when the mesh boundary is far from convex. In these cases, both schemes should be applied in a "smart" manner, namely, verifying that the triangles are still valid, or that the minimum angle of the adjacent triangles has been improved, before a vertex is moved. In some rare cases, both schemes may fail to improve the minimum angle when even Laplacian smoothing may improve it. A "combined" scheme that applies Laplacian smoothing when the angle-based method fails has extremely fast convergence and achieves the best of both worlds.

3.2 Area-based remeshing

The idea to use triangle areas as one of the criteria for triangulation optimization is not new. This usually means trying to form triangles with as uniform an area as possible. However, triangle areas alone cannot be used to obtain meshes of reasonable quality. The reason is that when only the areas are optimized, without taking into account the angles, the resulting mesh can (and in most cases will) have many long and skinny triangles. Only when a mesh has an almost regular connectivity may uniform triangle areas imply well-formed triangles. Nevertheless, a mesh containing triangles with uniform area distribution has one important positive property: The spatial distribution of the vertices over the total mesh area is very uniform. If we eliminate the edges of the mesh leaving only the vertices, we obtain quite a uniform point distribution, as may be seen in Figure 7(b).

We propose a remeshing scheme that utilizes this. Given a mesh, we alternate between the area equalization procedure and a series of angle-improving edge-flips. Edge-flips are performed until improvement is no longer possible. This process results in a mesh that is as close to regular as the ratio between the number of the boundary and interior vertices, together with the geometry of the boundary, allows. It is far superior to the results from an analogous scheme involving angle-based smoothing instead of area equalization. Figure 7(c) and (d) compare the two schemes.

To equalize the areas of the mesh triangles, a number of iterations are performed over the mesh. Each iteration moves all the mesh interior vertices sequentially to improve the areas locally. Let $p = (x, y)$ be an interior mesh vertex and $p_1 ... p_k$ its neighbors. (x_i, y_i) are the coordinates of p_i. Denote by $\mathcal{S}_i(x, y)$ the area of triangle (p_i, p_{i+1}, p). Note that $i + 1$ is modulo k:

$$\mathcal{S}_i(x, y) = \frac{1}{2}\begin{vmatrix} x_i & y_i & 1 \\ x_{i+1} & y_{i+1} & 1 \\ x & y & 1 \end{vmatrix}. \tag{3}$$

Let $\mathcal{S}$ be the area of the polygon $(p_i, \ldots, p_k)$ that is actually $\sum_{i=1}^{k} \mathcal{S}_i(0, 0)$. In order to find the position of p that equalizes the areas of the adjacent triangles as much as possible, we minimize the following function:

$$(x, y) = \arg\min_{(x,y)} \sum_{i=1}^{k} \left(\mathcal{S}_i(x, y) - \frac{\mathcal{S}}{k}\right)^2. \tag{4}$$

This reduces to solving a system of two linear equations in x and y. The computational cost of this unique solution is close to that of traditional Laplacian smoothing.

It turns out that a valid mesh can be obtained by equalizing the areas of the mesh triangles, even in cases such as a highly non-convex boundary. This contrasts with other methods, including the smart Laplacian [6] and both angle-based smoothing methods, which fail. See Figure 7(e)–(h).

	Min angle	Triangles < 10°	Triangles < 15°	Triangles < 20°
Laplacian	0.17°	2.57%	5.31%	8.71%
Angle-based	4.62°	0.58%	1.66%	4.56%
Weighted angle-based	17.2°	0.00%	0.00%	1.82%

Table 1: Quantitative comparison between quality of triangulations in Figure 6.

4. COMPATIBLE REMESHING

We now show how to combine the two methods introduced in Section 3, along with a refinement procedure (introducing new interior Steiner vertices), to produce high-quality compatible triangulations of two polygons given with a correspondence between their vertices. Compatible triangu-

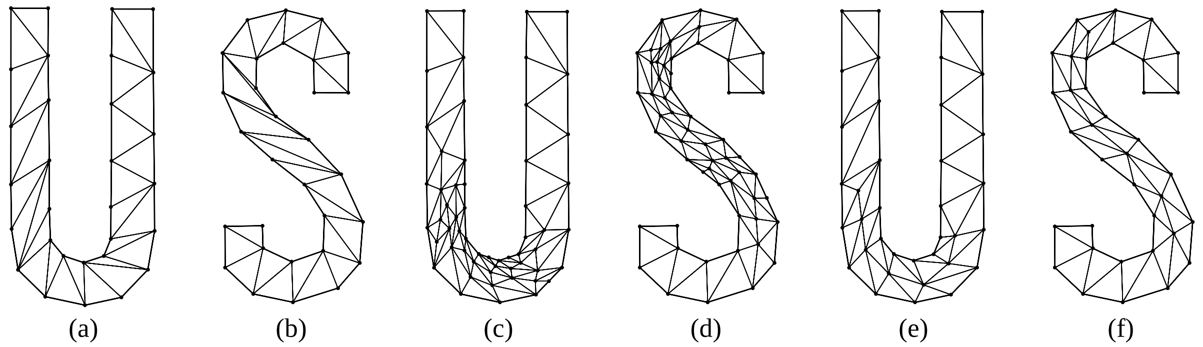


Figure 8: High-quality compatible triangulation of letters U and S. (a)–(b) Optimal compatible triangulations generated by the algorithm of Section 2. No Steiner vertices are required, but the minimum angle of U is 10.8°, and of S is 3.4°. (c)–(d) Compatible triangulations generated by the algorithm of Section 4, without area equalization. The number of Steiner vertices is 27, and the minimum angles are 15.4° and 15.7° respectively. (e)-(f) Compatible triangulations generated using area equalization. The number of Steiner vertices is 7, the minimum angles are 17.1° and 17.6°. The time required to generate (c),(d) and (e),(f) was similar.

lations created using the method introduced in Section 2 usually have a small number of Steiner vertices, but their quality is unlikely to be acceptable. Therefore, remeshing techniques must be applied to improve the quality. The main difficulty with using existing remeshing techniques is that the remeshing criteria which are suitable for a single mesh may fail when applied to two triangulations in parallel.

Our compatible remeshing technique is similar to that of Alexa et al. [1]. We use a series of simultaneous edge-flips, mesh smoothing and mesh refinement by edge-splitting. In addition, we perform a single iteration of the area equalization technique presented in Section 3.2. The outline of the algorithm appears in Algorithm 1. The parameter k_{split} dictates the rate at which new Steiner vertices are introduced.

```
while mesh quality has not been achieved or
      number of Steiner vertices does not exceed
      threshold
   Step 1.  Alternate between angle-based smooth-
            ing and simultaneous angle-improving
            edge-flips.
   Step 2.  Refine both meshes by $k_{split}$ simulta-
            neous edge-splits.
   Step 3.  The same as Step 1.
   Step 4.  Perform a single iteration of area
            equalization (Section 4).
   Step 5.  The same as Step 1.
```

Algorithm 1: Compatible remeshing.

While the criteria for operations in Algorithm 1 are rather straightforward for a single mesh, applying them simultaneously on two triangulations requires more precise control. If care is not exercised, the corresponding properties of triangles within the two meshes may often contradict each other. The following empirical criteria, based on their analog for a single mesh, have produced the best results on numerous examples:

Edge-flips: Similarly to when constructing Delaunay triangulations, the edge is flipped if the minimum angle between the angles of *both* meshes of the triangles adjacent to the edge is improved.

Angle-based smoothing: Both meshes are independently smoothed, applying the technique described in Section 3.1 in the “smart” manner, namely, preserving the validity of both meshes.

Edge-split refinement: Our criterion for choosing an edge e to be split is based both on the edge length (denoted by $|e|$) and the minimal of the four adjacent triangle angles ($\alpha_{\text{min}}(e)$). The edge with the maximal “normalized” length in both triangulations (T_0 and T_1) is refined:

$$e = \arg\max_{e \in T_0 \cup T_1} \frac{|e|}{(\alpha_{\text{min}}(e))^2}\,. \tag{5}$$

Note that the refinement is performed simultaneously on both triangulations in order to preserve the compatibility. The criterion defined in (5) produces better experimental results than the aspect ratio-based criterion of [12] or distortion metrics criteria of [4] and [6]. The number of edges to be split in each iteration (k_{split}) determines the trade-off between the number of Steiner vertices and the algorithm running time.

Area equalization: As noted in Section 3.2, area equalization improves the spatial vertex distribution. Due to the

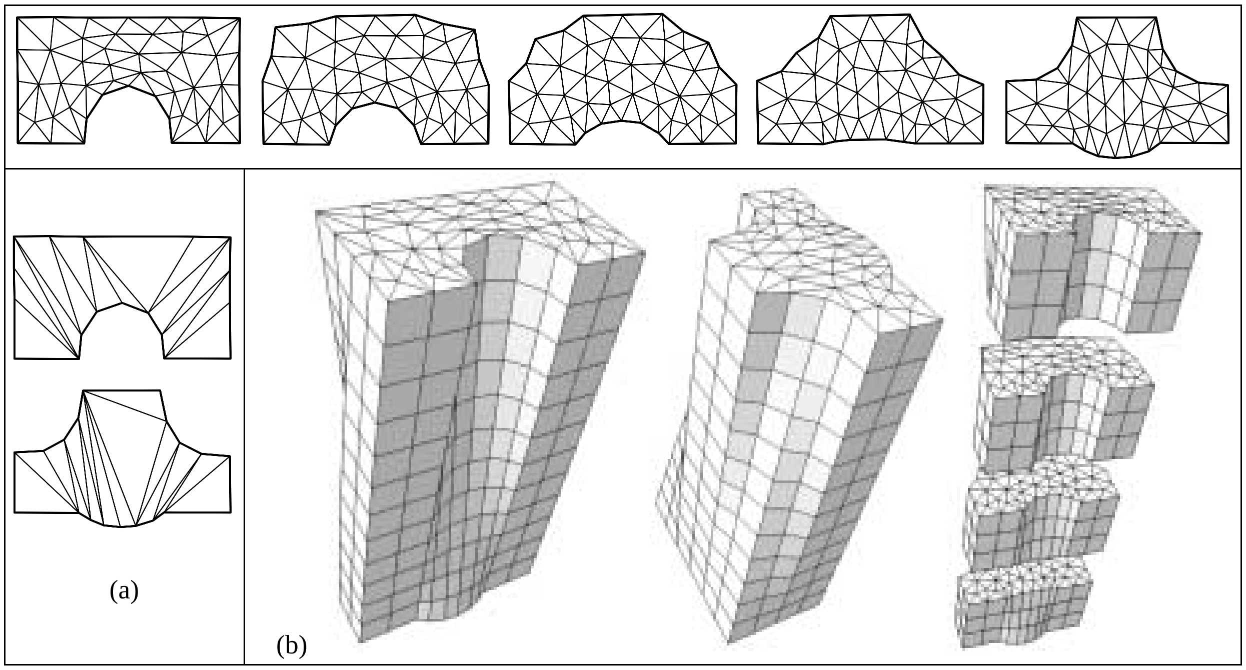


Figure 9: 3D sweep generation. (a) Optimal (no Steiner vertices) compatible triangulation of source and target polygons. Top row: High-quality compatible triangulation and intermediates generated by morphing procedure. Minimum angles of the source and target triangulations are 27.2° and 25.9°, respectively. (b) 3D visualization of sweeps from a number of different angles.

refinement operations, some regions of the mesh may have an excess in vertex density. To smooth this out, we apply a single iteration of area equalization (Step 4). This area equalization can prevent a further increase in the number of Steiner vertices at later stages, but at the price of slowing down the algorithm. See Figure 8. On the one hand, the refinement operations change the meshes locally, and thus, Step 1 (or 3) of Algorithm 1 converges quickly. On the other hand, the area equalization affects the mesh globally, and thus, Step 1 (or 3) takes much longer to improve the mesh globally. If a faster algorithm is required, Step 4 can be applied only every 4–10 iterations.

5. EXPERIMENTAL RESULTS

We have implemented all the algorithms described in this paper, and applied them to numerous example inputs. Our inputs consist of two planar polygons which serve as the source and target (top and bottom) cross-sections of the sweep. These two are compatibly triangulated with sufficient mesh quality (using Algorithm 1), and then morphed to create intermediate compatibly triangulated polygons. Especially challenging inputs are when the source and target are significantly different. Figures 9–11 show some sample input pairs, the (usually low-quality) compatible triangulations with a small number of Steiner vertices generated by the methods of Section 2, the remeshed high quality compatible triangulations generated by the methods of Section 4, and the intermediate triangulated cross-sections generated by applying morphing techniques. The latter are shown both as a sequence of 2D cross sections, and as a sliced 3D sweep. For each example, we specify the statistics of the source and target meshes. We found that the angles of the intermediate meshes generated using the techniques of Surazhsky and Gotsman [16], [17] were always in between those two, so the mesh quality is preserved throughout the morph.

In terms of runtimes, all these examples required no more than a second or so to run on a Athlon 1.2GHz PC with 256MB RAM. Larger inputs, which ultimately involved hundreds of (interior and exterior) Steiner vertices for the mesh and the morph, required no more than 10 seconds on the same machine.

6. DISCUSSION AND CONCLUSION

We have shown how to generate compatibly triangulated sweeps with quality adequate for finite-element analysis. Our method is fast, robust, and, as opposed to previously published methods, is guaranteed to always produce a valid result.

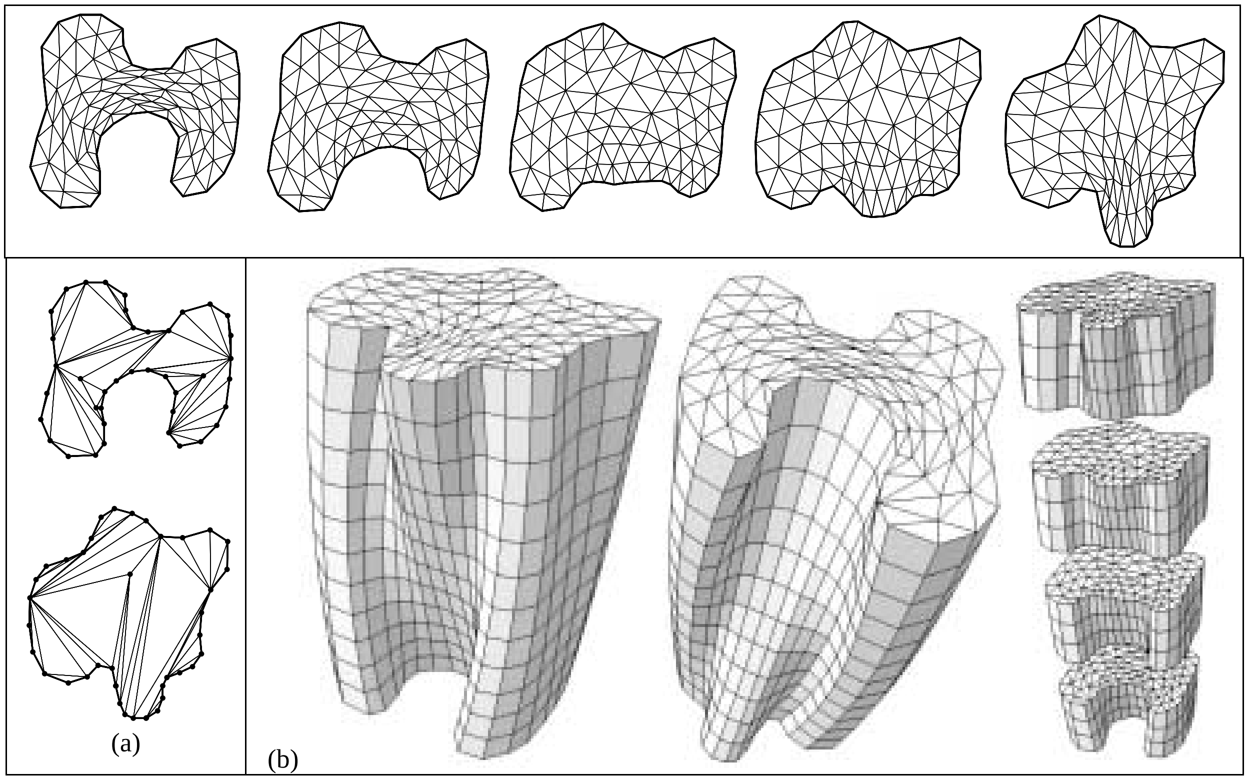


Figure 10: 3D sweep generation. (a) Compatible triangulation of source and target polygons with three Steiner vertices. Top row: High-quality compatible triangulation and intermediates generated by morphing procedure. Minimum angles of the source and target triangulations are 15.9° and 15.3°, respectively. (b) 3D visualization of sweeps from a number of different angles.

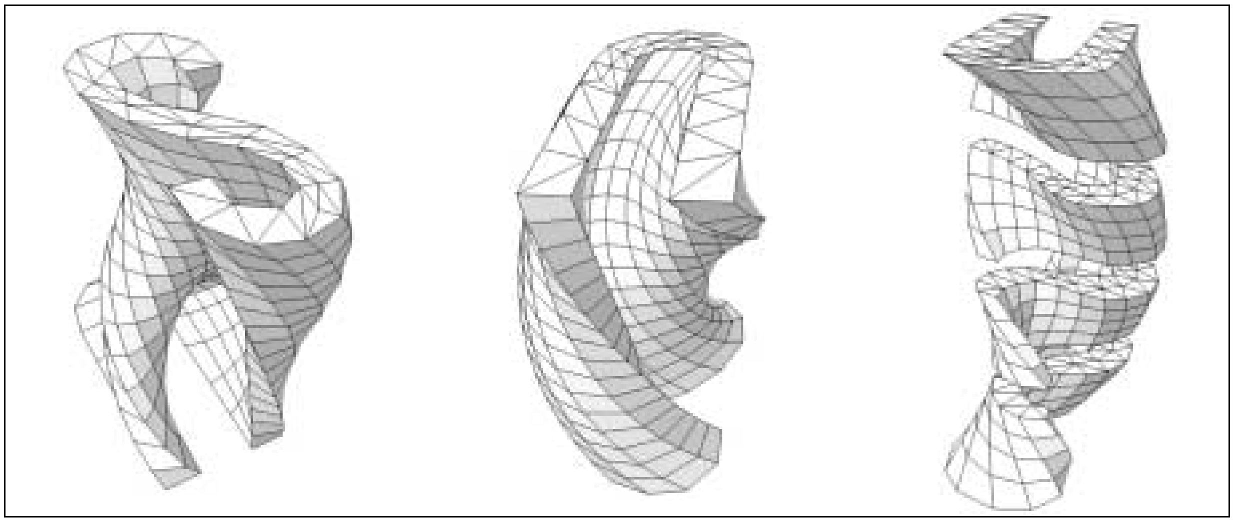


Figure 11: 3D visualizations of sweeps between letters U and S using compatible triangulations from Figure 8(e)-(f). The optimal compatible triangulations of U and S appear in Figure 8(a)-(b).

Several components of our algorithm, in particular the weighted angle-based smoothing procedure, may be used in their own right in other meshing applications.

The method was designed primarily for parallel planar inputs, but can probably be extended easily to the more general case. A sequence of source and target polygons, forming so-called "multi-sweeps" or "barrels" [5], may also be treated by decomposing the polygons.

Future work will address the case of hexahedral meshes.

ACKNOWLEDGEMENTS

Thanks to Tatiana Surazhsky and Michael Floater for their contribution to the area-based remeshing method, to Alla Sheffer for helpful discussions on sweeps, and to Gill Barequet for helpful discussions on the implementation of minimum-link path algorithms.

REFERENCES

[1] M. Alexa, D. Cohen-Or and D. Levin, *As-rigid-as-possible shape interpolation.* Proceedings of SIGGRAPH 2000, pp. 157–164.

[2] B. Aronov, R. Seidel, and D. L. Souvaine, *On compatible triangulations of simple polygons*, Computational Geometry: Theory and Applications, 3:27–35, 1993.

[3] M. Babikov, D. L. Souvaine, and R. Wenger, *Constructing piecewise linear homeomorphisms of polygons with holes*, Proceedings of 9th Canadian Conference on Computational Geometry, 1997.

[4] M. Berzins, *Mesh quality: A function of geometry, error estimates or both?,* 7th International Meshing Roundtable, pp. 229–238, 1998.

[5] T. Blacker. *The Cooper tool*. 5th International Meshing Roundtable, pp. 13–29, 1996.

[6] S. A. Canann, J. R. Tristano, and M. L. Staten, *An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes*, 7th International Meshing Rountable, pp. 479–496, 1998.

[7] M. S. Floater. *Parameterization and smooth approximation of surface triangulation*. Computer Aided Geometric Design, 14:231–250, 1997.

[8] M. S. Floater and C. Gotsman, *How to morph tilings injectively*, Journal of Computational and Applied Mathematics, 101:117–129, 1999.

[9] H. Gupta and R. Wenger, *Constructing piecewise linear homeomorphisms of simple polygons*, J. Algorithms, 22(1):142–157, 1997.

[10] P.M. Knupp, *Next Generation sweep tool: A method for generating all-hex meshes and two and one-half dimensional geometries*, 7th International Meshing Roundtable, pp. 505–514, 1998.

[11] E. Kranakis and J. Urrutia, *Isomorphic triangulations with small number of Steiner points*, International Journal of Computational Geometry and Applications, 9(2):171–180, 1999.

[12] V. Parthasarathy and S. Kodiyalam, *A constrained optimization approach to finite element mesh smoothing*, Finite Elements in Analysis and Design, 9:309–320, 1991.

[13] T. W. Sederberg, P. Gao, G. Wang, and H. Mu, *2D shape blending: An intrinsic solution to the vertex path problem*, Computer Graphics (SIGGRAPH '93), 27:15–18, 1993.

[14] M. Shapira and A. Rappoport, *Shape blending using the star-skeleton representation*, IEEE Trans. on Computer Graphics and Applications, 15:44–51, 1995.

[15] M. L. Staten, S. A. Canann and S. J. Owen, *BMSweep: Locating interior nodes during sweeping*, 7th International Meshing Roundtable, pp. 7–18, 1998.

[16] V. Surazhsky and C. Gotsman, *Controllable morphing of compatible planar triangulations*, ACM Transactions on Graphics, 20(4):203–231, 2001.

[17] V. Surazhsky and C. Gotsman, *Guaranteed intersection-free polygon morphing.* Computers and Graphics, 25(1):67–75, 2001.

[18] S. Suri, *A linear time algorithm for minimum link paths inside a simple polygon*. Comput. Vision Graph. Image Process., 35:99–110, 1986.

[19] S. Suri, *On some link distance problems in a simple polygon*, IEEE Journal of Robotics and Automation, 6(1):108–113, 1990.

[20] W. T. Tutte, *How to draw a graph*, Proc. London Math. Soc., 13:743–768, 1963.

[21] T. Zhou and K. Shimada, *An angle-based approach to two-dimensional mesh smoothing*, 9th International Meshing Roundtable, pp. 373–384, 2000.