The Visual Computer manuscript No.
(will be inserted by the editor)

Joe Kahlert · Matt Olson · Hao Zhang

Width-Bounded Geodesic Strips for Surface Tiling

Abstract We present an algorithm for computing families of geodesic curves over an open mesh patch to partition the patch into strip-like segments. Specifically, the segments can be well approximated using strips obtained by trimming long, rectangular pieces of material having a prescribed width δ. We call this the *width-bounded geodesic strip tiling* of a curved surface, a problem with practical applications such as the surfacing of curved roofs. The strips are said to be straight since they are constrained to fit within rectangles of width δ, in contrast to arbitrary, possible highly curved, strip segments. The straightness criterion, as well as a bound on strip widths, distinguishes our problem from ones previously studied for developable surface decomposition.

We start with a geodesic curve defined by a user-specified starting point and direction on the input mesh. We then iteratively compute the neighboring geodesics which respect the constraints and lead to optimal use of material (i.e., minimizing the trimmed material) until the surface mesh in question is completely tiled. Our algorithm is exact with respect to the polyhedral geometry of the mesh surface and it runs on a variety of surfaces with a modest time complexity of $O(n^{1.5})$ under reasonable input parameter settings, where n is the mesh size. We also show how the algorithm can be extended by relaxing the constraint that neighboring geodesics span the mesh, thus allowing the algorithm to be applied to meshes with greater undulation.

Keywords geodesics · surface tiling · straight strips

1 Introduction

Anyone who has made a *papier-mâché* model knows that a curved surface can be formed using strips of flexible material. If these paper strips are trimmed to eliminate overlap, the result is a *strip tiling*. In industrial settings, whether it be for building shells, ship hulls or timber construction, material is typically available in long rectangular strips or spools. Such strips can be cut, trimmed and joined together to approximate a surface.

Graphics, Usability, and Visualization (GrUVi) Lab
School of Computing Science
Simon Fraser University, Canada

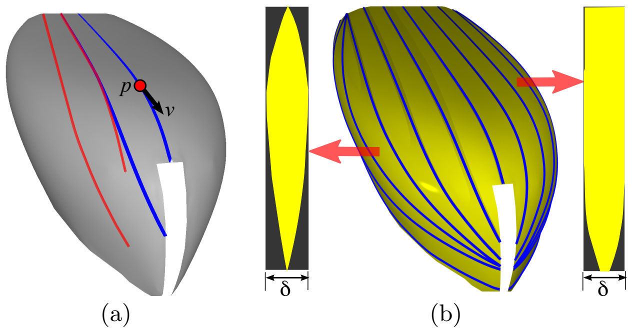


Fig. 1 Width-bounded geodesic strip tiling of a ship hull surface. (a) User input includes a point p and an orientation v, resulting in the initial geodesic through p. An intermediate processing step showing the geodesic found in the first iteration (second blue curve), along with the upper bound curves (in red) which constrain the search for the geodesics, is shown as well. (b) The complete tiling and two resulting strip widths plotted over-top of rectangular material pieces showing that the δ width constraint is satisfied.

In this paper, we are interested in the following *straight strip tiling* problem: given a curved surface patch without holes, approximate it using an adjoining set of straight strips satisfying a bounded-width constraint. Specifically, each strip must fit within a long rectangular piece of material with a given width δ; see Figure 1. In other words, the tiling strip can be obtained from the rectangular piece by cutting and trimming which for our purposes makes it straight. Note here that there is no constraint on the length of the rectangles.

In a practical application such as roof building in Figure 2, we can directly observe the need for *straight* tiling as the long metal roof panels used flex (to any significant extent) in only one direction — perpendicular to their length and normal to their surface. In other words, these panels remain *straight* when applied, so the technique described herein is needed to prescribe their

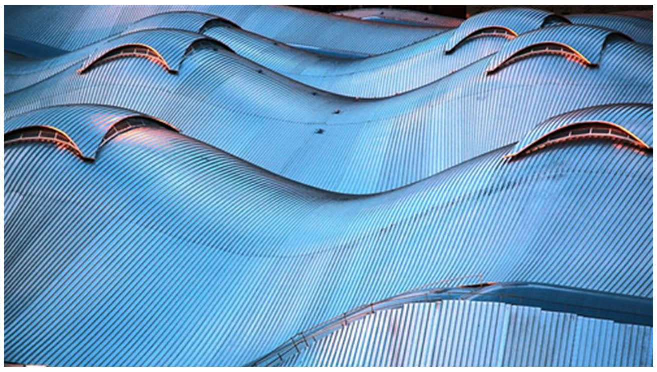


Fig. 2 The Southern Cross Station in Melbourne, Australia covered with straight metal strips (from flickr.com).

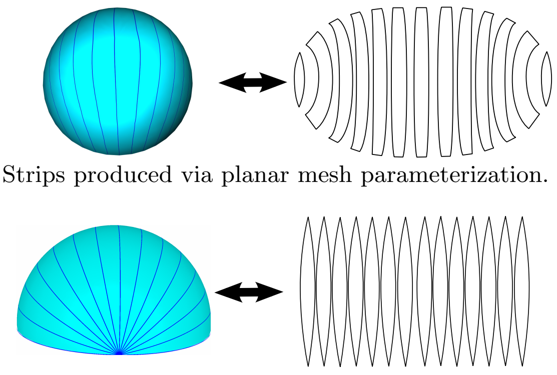

Strips produced via planar mesh parameterization.
Strips produced via width-bounded geodesic marching.

Fig. 3 Tiling a hemisphere patch using strips. Top: Strips produced by straight lines drawn over a conventional planar parameterization of the patch are not straight. Bottom: Straight strips derived from our width-bounded geodesic marching algorithm.

cutting and placement necessary for construction. Needless to say, reducing material waste is also of paramount practical importance here.

A naive solution to this problem would be polygonal meshing, tiling the input surface using planar facets each of which is sufficiently small to fit within the material width. However, this does not exploit the long length of the material and thus leads to more cutting and joining than is necessary. Planar mesh parameterization is also an option, whereby the given surface is first flattened so as to minimize distortion and then straight lines in the parameter domain are mapped back onto the original surface to define the strip boundaries. While excellent algorithms for mesh parameterization exist [22], they do not provide satisfactory solutions with respect to the straightness criterion. Consider for example the case of a hemispherical mesh. Conventional mesh parameterization schemes would flatten the input surface into a planar circular patch while distributing distortion uniformly across. The resulting strips on the hemisphere, obtained using curves corresponding to straight line boundaries in the parameter domain, when approximately developed, are curved in that they do not fit within our rectangles of width δ (unless δ is very large — on the order of the mesh size — which is of little interest here as we seek an algorithm that works for *any* δ). See Figure 3 for an illustration of such a situation, comparing with straight strips derived from our approach.

Recent work of Pottmann et al. [20] takes as input a set of curves on a mesh surface and produces a set of adjoining developable strips (constrained by the input curves) which together provide a good surface approximation. If one of these curve families consists of *geodesics*, then this algorithm can be adapted to produce strips with *approximately straight development*. Making use of this work affords us the luxury of addressing our strip tiling problem by only focusing on covering the surface with geodesics and then deferring to Pottmann et al. [20] for constructing the actual material strips to approximate the surface. Specifically, we frame the problem we wish to solve as follows: on a given mesh patch, find a set of nonintersecting geodesic curves that are separated by no more than the prescribed material width δ (measured geodesically) from each other and from the surface boundary so that the resulting strips, when developed, span the width of the bounded-width rectangles as much as possible so as to minimize material waste.

Finding a globally optimal solution to such a problem is beyond the scope of this paper. Instead, we solve a greedy version which is initialized with a point and an orientation on the surface. Given this input provided by the user, we can launch an initial source geodesic curve from which we can iteratively find neighboring geodesics with the properties mentioned until the surface is covered with geodesic strips of bounded width. We call such a procedure *width-bounded geodesic marching*; refer to Figure 1 for an illustration on a ship hull surface.

1.1 Motivation

It is well known that when a narrow strip of flexible yet unstretchable material is placed on a smooth surface to conform to it, the strip will follow a geodesic path [18]. This results from the *straightness* property of geodesics — a geodesic over a smooth surface curves only in the direction of the surface normal — and it motivates the use of geodesic marching for straight strip tiling of surfaces. With the recent work of Pottmann et al. [20] which can convert sets of geodesic curves on a surface into adjoining strips to approximate the surface, we are left to address the problem of computing an optimal set of geodesics fulfilling the various constraints.

Our geodesic-finding algorithm draws upon much recent excellent work on geodesic computation over discrete surfaces [4, 13, 23]. Specifically, we make significant use of the commonly adopted concept of "windows" for geodesic marching. However, previous works focused on solving the *shortest* path problem to find geodesics and while all shortest paths are indeed geodesics, not all geodesics are shortest paths [8]. Therefore, in order to find *all* of the geodesics necessary for our strip tiling problem, we need to extend these existing works.

If material strips need not be straight when unrolled, or *developed*, into the plane or if the surface to be tiled is rather special, e.g., section of a sphere, developable surface, or surface of revolution, then either existing techniques (e.g., stripification) or surface properties (e.g., symmetry) may be exploited to compute the geodesic curves covering the surface. However, if only planar material available in *rectangular strips of bounded width* may be used to cover an *arbitrary curved surface*, then to the best of our knowledge, our algorithm is the only available solution so far.

1.2 Applications

Roof construction With the exception of single-layered roofs made from exotic materials such as glass or fabric, most building roofs have two distinct components: the supporting structure and weather resistant surfacing. For curved roofs that have been gaining popularity in modern architecture, construction of both of these components is far more complicated than for the case of planar roofs. For curved roof surfacing the de facto standard materials are metal roofing panels which are abundantly available in long straight strips. Figure 2 shows a large curved roof covered with metal panels trimmed, placed and joined to tile the roof surface. Given a mesh of this surface and the material width, our algorithm (in conjunction with a strip construction algorithm like Pottmann [20]) can be used to prescribe the trimming and placement of the material strips necessary to panel the roof.

Boat hull construction Some boat hulls are made from straight flexible materials such as thin wooden strips. Figure 1 shows an example of how our algorithm can be used to produce the strips necessary to form the hull.

1.3 Contributions

Our contributions include the following. First, we adapt existing shortest-path techniques in order to be able to find *straight*, as opposed to shortest, geodesics. Then, we develop techniques to satisfy more geodesic curve constraints of greater complexity from a wider range of starting points as compared to the constraints posed by shortest-path problems. We present a novel geodesic area calculation technique which enables us to choose strip candidates that minimize material waste. Finally, we allow straight strip tiling of a wider variety of meshes, including those with greater undulation, by relaxing the constraint that neighboring geodesics span the mesh. In other words, the delimiting geodesics constructed are allowed to intersect in the interior of the input patch.

Our algorithm is exact with respect to the polyhedral geometry of the mesh surface and it runs on a variety of surfaces with a modest time complexity of $O(n^{1.5})$ under typical input parameter settings, where n is the mesh size. The required user-defined orientation typically allows us to incorporate important property of the surface or application-specific knowledge, e.g., symmetry, gravity, or other considerations, into the algorithm.

2 Related work

In this section, we discuss possible alternative approaches to solving our straight strip tiling problem and techniques upon which we build in the course of developing our chosen approach.

2.1 Meshing and stripification

Technically speaking, our stripping tiling problem can be solved by remeshing [2] the input surface with small enough planar facets. However this solution does not take advantage of the available length of the material strips and thus leads to an unnecessarily fractured result.

Recent work from architectural geometry [11] uses planar quad meshes to represent free-form structures. This work is then extended by combining these quadrilaterals into continuous singly curved strips in what is known as a *semi-discrete surface* representation [20]. Pottmann et al. go on to show that if geodesic curve families are used as input then straight strips which approximate the input surface can be produced. In a similar work [21] the same authors call for the creation of curve networks to initialize their semi-discrete surface optimizations which is in fact the objective of this paper. If we are successful in finding such a family of geodesic curves then this existing body of work can be used to approximate the input mesh with a semi-discrete surface made out of singly curved strips that are straight when unrolled.

There are also a variety of mesh stripification algorithms [24]. The common issue with them is that they do not address the straightness criterion. Our effort is on finding straight geodesic curves on an open mesh patch satisfying user and material width constraints

2.2 Discrete geodesics

Geodesics are most well known for the fact that the shortest path between two points on a smooth manifold is guaranteed to be geodesic [3,5]. However, they are more rigorously defined as curves whose curvature vector is parallel to the surface normal. There have been many studies extending this concept of differential geodesics from smooth surfaces to discrete geodesics on polyhedral surfaces that are represented by meshes.

Shortest geodesics are defined as locally shortest curves on a mesh [6,13]. The most popular method for finding these shortest geodesics on meshes has been using fast marching to solve the Eikonal equation [10]. Using this method, distances from a source point are propagated across the mesh face by face with shorter distances canceling out longer ones.

The other popular approach for finding shortest paths exploits their *straightness* property. The MMP algorithm of Mitchell et al. [13], originally introduced in 1987 and made practical by Surazhsky et al. in 2005 [23], divides the mesh into intervals — or *windows* — without vertices over which geodesics exist simply as straight lines in that window's planar unfolding. By creating, propagating and merging appropriately, *straight* windows can be extended from any source point to all other points on the mesh and the shortest path can simply be selected from among those that reach the intended target. Bommes and Kobbelt [4] extend this work for geodesic marching to polygonal segment sources rather than just single points. Liu et al [12] warn of *degeneracies* that exist in the Surazhsky algorithm.

Straightest geodesics are those whose left and right curve angles (the sums of the incident angles on either side of a point on the curve on a mesh) are always equal [17]. Straightest geodesics have application to the translation of vectors and the integration of vector fields and are related to shortest geodesics in the following ways [17]:

- A geodesic containing no surface vertex is both shortest and straightest.
- A straightest geodesic through a *spherical* vertex is not locally shortest.
- There exists a family of shortest geodesics through a *saddle* vertex. Only one of them is a straightest geodesic.

Note that we define saddle and spherical vertices in Section 3.2, since they are relevant to the discussion of our algorithm.

Quasi-geodesics Despite their popularity, neither of the above discrete geodesics serves our purpose since we are interested in *all straight geodesics* — not just those that are the shortest or the straightest. Quasi-geodesics are the limit sets of smooth geodesics on polyhedral surfaces [1,15]. They behave identically to shortest geodesics with the exception that there is also a family of quasi-geodesics through a spherical vertex, which cannot contain shortest geodesics [13]. It is this higher adherence to the differential version of the geodesic that makes quasi-geodesics the most suitable for our application.

2.3 Mesh parameterization

Mesh parameterization techniques [7,22] also offer potential. Specifically, the straight strip tiling problem could be solved by a parameterization that preserved straightness in the sense that straight lines in the parameter domain would correspond to geodesics on the original surface (and vice versa). However, we are not aware of any existing algorithm that possesses this property in general. On a hemisphere mesh patch, conventional parameterization algorithms, i.e, angle-preserving approaches such as angle-based flattening [22], would lead to curved strips, as illustrated in Figure 3.

In fact, a closer look at the example of the hemisphere suggests that there may be no *one* mesh parameterization that preserves straightness *in all directions*. We can see this by first observing that the only geodesics that produce straight strips on a hemisphere are the so-called great arcs which must start and end at the same two poles to avoid intersection (see Figure 3). In other words, these two singularities are inevitable for any straightness-preserving flattening procedure. However, a parameterization with such isolines would clearly not preserve straightness in *any other* direction as the resulting geodesics would converge on two different poles. Therefore for any given surface we would require a family of parameterizations — each one corresponding to a particular strip alignment — to solve the problem in general.

2.4 Geodesic flow

Another approach, that of the *geodesic flow* [14], also offers potential for solving the straight strip tiling problem. A geodesic flow is a tangential vector field on a surface whereby all resulting integral curves, i.e., curves whose tangents vectors belong to the vector field, are geodesic. If a geodesic flow for a surface could be found, then the desired strip boundaries for any material width could be found trivially. There has been some work on geodesic flows for discrete surfaces [16] describing the properties of fields corresponding to geodesics that emanate from a single point on a mesh. However, application of this theoretical technique to our problem appears to be difficult, hence we have chosen not to pursue it here.

3 Width-bounded geodesic marching algorithm

As mentioned in Section 1, the work of Pottmann et al. [20] allows us to concentrate exclusively on the creation of geodesic curves rather than the construction of 2D strips from these curves. Our goal then is to seek a family of curves on a given open mesh patch so that the following constraints or criteria are satisfied:

- **Geodesic:** The curves must be geodesics.

– **User constraint:** The initial source geodesic must respect the user-specified point and orientation.
– **Width constraint:** Given that we expect our flexible material to conform to the surface, no point on any curve should be more than the material width δ geodesic distance away from either of its neighboring curves. In other words, a "geodesic circle" with radius δ centered at any point on any curve should *intersect* another curve or the mesh boundary on either side; see Figure 4.
– **Best use of material:** The found curves should be as far away as possible from each other.
– **Intersection-free:** The curves should not intersect in the patch interior.

Note that there is no guarantee that all the constraints above can be satisfied for any surface with any width bound δ. If we can find such a family of geodesic curves, then we can pass them to the strip generation algorithm of Pottmann et al. [20] to produce the straight strips to tile the input mesh.

However when the input surface is sufficiently curved with a sufficiently small δ bound, the last constraint above can be impossible to satisfy. In Section 3.5, we present an extension to our core algorithm which allows long strips to be "severed" when the constructed geodesics intersect in the patch interior. This enables us to handle surfaces with greater undulation. However, since the resulting curve families now contain intersections, they cannot be passed as input to Pottmann's algorithm [20] for strip production — extensions of this technique are required to support severed strips.

3.1 Initial source geodesic curve

We first create the initial curve from the user-specified surface point p and tangent vector $\mathbf{v}$ to serve as the source. Since our curve is to be geodesic, we can simply start at the given point and proceed "straight" along the surface in the given direction until we reach the boundary. To do this, we utilize the classical definition of a geodesic as a path whose osculating plane also contains the surface normal [5]. In the discrete version, a geodesic proceeds straight over faces and preserves incident angles when traveling over edges [9]; traveling directly over vertices will be discussed below in Section 3.2. Using this simple rule we implement a *geodesic walk* that takes as input a starting point and tangent vector and simply "walks" forward producing a geodesic curve as output.

3.2 Constrained neighbor geodesic construction

Given this source geodesic, we implement a method for finding a neighboring geodesic that satisfies the constraints mentioned above. Then we use this method iteratively,

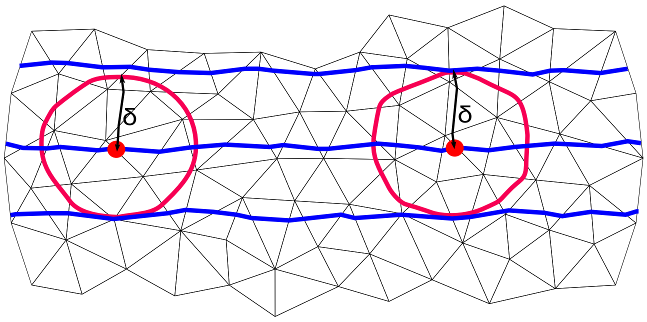


Fig. 4 Geodesic circles of radius δ illustrating the width constraint test for two points (red dots) on the middle curve. The point on the right passes the test since the circle intersects another curve on either side while the point on the left does not. Every point on every curve must pass this test.

with the newly found neighbor geodesic serving as the source, to find subsequent neighbors until we reach the boundary of the mesh patch. Finally, we repeat this iterative process on the other side of the source to find the remaining set of geodesics to cover the entire surface.

Upper bound curve To satisfy the width constraint, we first construct an *upper bound curve* based on the current source geodesic and use it to delimit the search for a neighbor geodesic. To this end, we appeal to the work of Bommes and Kobbelt [4] which enables us to find the boundary of the region locally "swept out" by our geodesic circle of Figure 4 as its center travels along the source curve. This resulting connected curve is an upper bound curve in the sense that it represents the furthest extent of all of our geodesic boundary circles.

We can use this upper bound curve to simultaneously enforce the width constraint at all points along a source geodesic by requiring that the neighbor geodesic not cross the upper bound. Also, since our upper bound is unbroken the bounding constraint is reciprocal: if a neighbor geodesic $\hat{c}$ to the source geodesic c respects the upper bound constructed from c, then c also respects the upper bound curve constructed from $\hat{c}$ using the same width constraint.

Vertices (spherical vs. saddle) and windows Given a current source geodesic, the upper bound curve can be obtained by constructing a set of *vertex-free regions* called *windows* extending perpendicularly from the source curve. We do this because mesh regions without vertices can be *unfolded* or flattened without distortion onto the plane where all geodesics then exist as straight lines. It is for this reason that windows are used in geodesic computations [4,13,23]. The intuition behind our construction of the upper bound curve via windows is that we are unfolding the mesh from the source geodesic toward the upper bound curve.

Note that despite our alternative definition these windows are the same as those used in previous work [4,13,

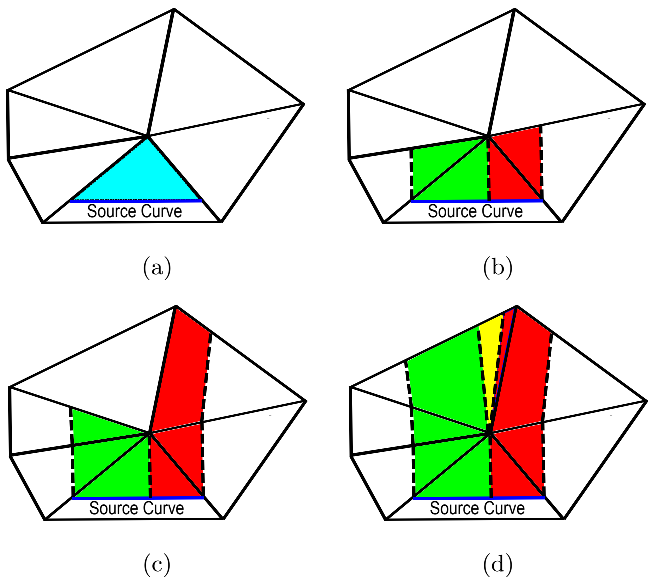
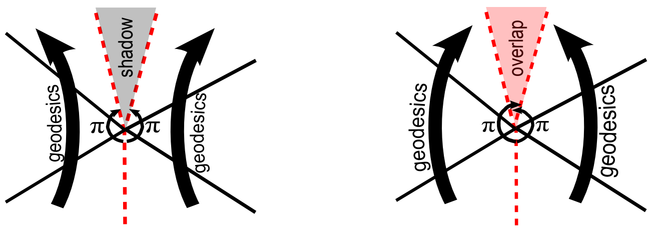


Fig. 5 Window *regions* each of which contains a continuous "side-by-side" set of geodesics emanating in a perpendicular direction from the source geodesic curve. Here a *linear window* propagates first in the planar face containing the source (cyan window) in (a), splits (into red and green) around a vertex in (b) which continue in (c) and finally give rise to a *vertex window* (yellow) in (d). The vertex shown is a *saddle vertex* (Figure 6) as is evidenced by the incident window separating and leaving a gap filled by the circular vertex window (yellow).



(a) Saddle vertex: windows leave a "gap". (b) Spherical vertex: windows overlap.

Fig. 6 Saddle (a) and spherical (b) vertices and the window boundaries surrounding them. A vertex splits the incoming window into two windows and gives rise to a third window emanating from this new or *pseudo-source*.

23] in which they are defined as a set of *intervals* on edges propagated across the mesh. In contrast we talk about window *regions* which we define as the set of all points on the mesh traversed by all geodesics within a window *interval* as these geodesics travel back to the source. In other words, an interval corresponds to a continuous set of adjacent geodesics all lying within a bounded region of the mesh — see Figure 5 for an illustration.

The unfolding process is complicated by the existence of mesh vertices which inevitably introduce distortions. Indeed, the behavior of geodesics near mesh vertices requires special care. To this end, one makes distinction between two types of vertices:

- **Spherical (elliptical) vertices:** a mesh vertex where the sum of face angles incident at the vertex is less than or equal to 2π.
- **Saddle (hyperbolic) vertices:** a mesh vertex for which the face angle sum is greater than 2π.

Recall that we are interested in all straight geodesics. When a geodesic ray (a straight curve on the mesh) passes through a vertex and extends *straightly* beyond that vertex, it may continue along *any* ray between the two rays that are at an angle π from the incident ray [23] and still retain its (straight) geodesic properties. In Figure 6, the ray can be extended anywhere within a region marked either as a *shadow* or an *overlap*. The distinction between the two cases dictates how windows are constructed near the vertices as explained below. There are two types of windows [4]:

- **Linear window:** A linear window contains geodesics emanating in a perpendicular direction from a segment of our source curve. In the unfolding of this window's faces the boundary of the region swept out by our geodesic circle is easily computed as a line segment that is parallel to the source. A linear window is shown propagating in Figure 5 and is marked in cyan in Figure 7.
- **Circular window:** A circular window contains all geodesics emanating from a single point on the surface. While we do not require circular windows to construct our initial upper bound, as linear windows propagate they split at vertices (Figure 5) giving rise to shadows and overlaps which contain circular windows propagating radially from these new or *pseudo*-sources. Figures 5 and 7 both show circular windows emanating from pseudo-sources in yellow. In the unfolding of a circular window the extent of our geodesic circle is simply a circular arc.

Note that our need to create circular windows for both saddle and spherical vertices arises from our need to use quasi-geodesics. The resulting implementation differences from previous work on finding discrete geodesics [4, 23] are described in Section 4. This difference also simplifies area calculations as described in Section 3.3.

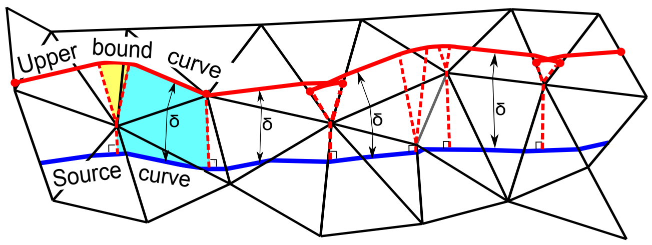


Fig. 7 Linear (an example in cyan) and vertex (an example in yellow) windows (delimited by dashed lines) constructed from the current source geodesic (blue). The width constraint is given by δ. The upper bound curve (red) is made up of straight (i.e. geodesic) and circular arc segments.

Construction and use of upper bound The geodesic and arc segments in our windows above join together to form our upper bound curve. Figure 7 shows the upper bound curve as a connected set of these segments. As mentioned, in order to ensure that the width test passes at all points on the source and the neighbor geodesic, we need only ensure that our neighboring geodesic does not cross the upper bound curve. Furthermore, we can interpret our constraint that neighboring curves be as far away as possible from each other to mean that at least one point on the neighboring geodesic must *touch* the upper bound. If this were not the case, we would greedily claim that our neighbor would not be far enough away from the source and thus we would not be making the best use of material. The possibility that a neighbor not touching the upper bound could in fact be the optimal choice seems unlikely; however, formal confirmation of this fact is left to future work.

Construction of neighbor geodesic With the knowledge that our neighbor geodesic must touch, but not cross, the upper bound, we consider the upper bound curve's shape. If we unfold the faces local to a point along the upper bound curve on the surface, it corresponds to a point on a polygon in the plane. Furthermore the problem of finding our noncrossing geodesic becomes the problem of finding a line segment touching — but not crossing — our planar polygonal upper bound. Under this useful transformation we can see that any line segment that touches but does not immediately cross over the polygon must touch a point where the polygon *bends away* from the line segment. Thus we can label all such points on the upper bound curve that bend away from the source as *convex corners* (green dots in Figures 7 and 8) and conclude that all valid candidate geodesics *must touch at least one convex corner* of the upper bound.

We can then use this fact to enumerate all potential candidate geodesics by observing that all geodesics touching one convex corner must lie within double-sided angular ranges, or "butterflies", emanating from this convex corner, as shown in Figure 8. We find all the geodesics within each butterfly (i.e. those traveling *through* our convex corner) by propagating a pair of circular windows in opposite directions and then pairing up the resulting windows that are directly across from each other (splitting if necessary). Finally, in order to satisfy the width and intersection-free constraints, we discard any of portions of any windows that cross back over the source or upper bound curves.

3.3 Optimizing material usage

While different applications may have unique criteria against which to judge and select the best valid candidate geodesics, one that is likely to have universal utility is that of material usage maximization. Specifically,

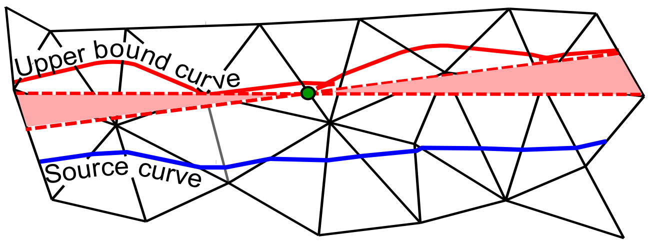


Fig. 8 A "butterfly" window: a pair of opposing angular ranges (delimited by the dash lines) emanating from a *convex corner* point (green dot). A convex corner is a point along the upper bound curve where it curves "away" from the source; more convex corners are marked in Figure 7, also as green dots. The "butterfly" window contains all the candidate geodesics touching that convex corner.

among all the valid candidates in one iterative step, we greedily select the one which results in the most *usage*, measured as the enclosed surface area divided by the length of the source curve.

The enclosed area between the source and a candidate geodesic curve can be found by summing the areas of the enclosed faces. However, in order to select the single geodesic enclosing the greatest area from amongst a set of windows each containing continuous range (and therefore infinite number) of geodesics we must consider the geometry of the windows themselves. Specifically, in the planar development — or unfolding — of a window's faces (an area preserving transformation), geodesics become *straight lines* and the regions between geodesics emanating from the same source become thin planar triangles (Figure 9). Moreover, if successive windows are alongside (i.e. in full contact with) one another, we can propagate area calculations to the next window. Finally, we can even propagate area calculations from butterfly to butterfly by exploiting the fact that the last geodesic of one butterfly is actually the first geodesic of the next. Thus, we can propagate our initial brute-force face summation by simply adding window areas (and subtracting overlap areas) as we step sequentially from window to window and from butterfly to butterfly.

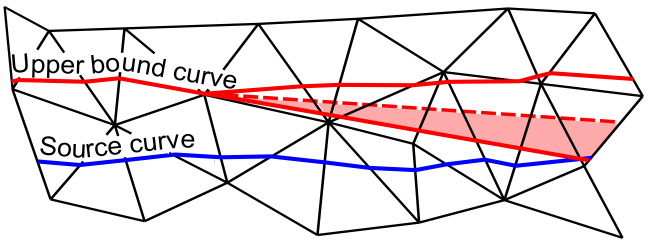


Fig. 9 Incremental computation of enclosed areas. Given the area enclosed by one geodesic (red dashed line) in a window (delimited by solid red lines), the area enclosed by any other geodesic in that window can be found by adding or subtracting the triangle between them (shaded) in the unfolding of the window's faces.

Fortunately, since we are using quasi-geodesics, the windows emanating from a source point are indeed always alongside each other. This follows from the property that quasi-geodesics travel across spherical vertices similarly to saddle vertices in that geodesics incident upon them may continue along any ray between the two rays at an angle π from the incident (Figure 6). This is opposed to previous works on finding discrete geodesics [4,23] in which spherical vertices are ignored as shortest paths never travel directly over them [13]. These spherical vertex windows ("overlap" region in Figure 6(b)) bridge the "gap" between the windows on either side of the vertex ensuring that all windows are alongside one another. In Section 4, we show how to extend works on discrete (shortest) geodesics [4,23] to find quasi-geodesics.

3.4 Computational complexity

Let F, E, and V be the number of faces, edges, and vertices in the input mesh and let $n = \max(F, E, V)$. In the first stage of the algorithm the initial geodesic is constructed using the geodesic walk algorithm described in Section 3.1. To do so the algorithm must "walk" over $O(F)$ faces (with each face requiring constant time) for each of the geodesic curves that are to be output.

The second stage of the algorithm involves the iterative search for a neighboring geodesic given a source geodesic. This stage includes the following steps:

1. Construction of the upper bound curve by extending windows from the source geodesic (Figure 7) including the construction of $O(F+V)$ (i.e. linear and vertex) windows each propagating over $O(F)$ faces
2. Creating butterfly windows at $O(V)$ convex corners of the upper bound (Figure 8) and searching each for area optimizing geodesics (Figure 9) using $O(V)$ windows each containing $O(F)$ faces

Step 2 dominates resulting in a worst-case complexity of $O(n^3)$. Note that the area calculation described in Section 3.3 requires only constant time for each window and thus does contribute to the asymptotic complexity analysis. Also, we do not require the $O(\log V)$ steps needed to perform the window sorting required by shortest-path algorithms [23], as we explain in Section 4.

Now let us take a more practical view on the computational complexity. On a sufficiently smooth mesh, the butterfly windows from Step 2 do not overlap appreciably, which means that there should be approximately $O(V)$, instead of $O(V^2)$, windows within all of the butterflies over the entire mesh. Moreover, each of these windows need propagate over only the width of the patch which should only be $O(\sqrt{F})$, instead of $O(F)$, faces on a uniform mesh that is reasonably close to being square (i.e., its width and length are of the same order).

Under these simplifying assumptions, the complexity reduces to $O(n^{1.5})$. It is important to note here that this argument does not constitute a rigorous average-case complexity analysis but rather a best guess at the expected performance of our algorithm under what we consider as reasonable input conditions: smooth, uniformly tessellated, and square meshes with a material width that is not much smaller than the average mesh edge length. Note also that smoothness and uniformity of tessellation can always be achieved by preprocessing of the input mesh.

3.5 Strip severing

The algorithm described thus far attempts to cover an open mesh with nonintersecting geodesics. For highly curved surfaces, this is not always possible. However if we allow the constructed geodesics to touch each other, we can then remove this limitation and extend the algorithm's applicability to any surface on which a valid source geodesic can be found. That is, if a source geodesic starting from and ending on the mesh boundary that does not intersect itself can be found, then using this technique our algorithm will succeed.

This amounts to relaxing the constraint that candidate geodesics do not intersect the source. That is, candidate geodesics with one or both endpoints on the source geodesic are now permitted. When we select one of these segment candidates as our greedy choice, our source curve in the next iteration will be a connected set of geodesic segments rather than a single geodesic curve. The corresponding upper bound curve will simply be the union of each geodesic segment's upper bound. They should always meet due to the fact that these geodesic segments must always join at convex angles, but we do not have a formal proof of this claim as it is left for future work. In practice, this technique has always worked as expected and we demonstrate these results in Section 5. In particular, Figure 11 (b), (c) and (d) show many strips severed by other strips meeting at junctions of arbitrary angle.

4 Window propagation

The implementation of our algorithm follows closely the published works of Surazhsky et al. [23] and Bommes and Kobbelt [4]. In this section, we provide details on *window propagation*, which form the most notable differences from and additions to the previous works.

Window propagation is the process of constructing windows starting from a source point or line segment and preceding forward face by face. In our implementation, windows are propagated from the source curve to form the upper bound (Figure 5) as well as from butterfly windows to find candidate neighbor geodesic curves.

The objective of both of these window propagations is to find *all* quasi-geodesics. This differs from previous works [4,23] that seek only the shortest geodesics. As such, our implementation contains a number of differences and extensions.

Spherical Vertices As mentioned in Sections 3.2 and 3.3, instead of being ignored, spherical vertices give rise to windows emanating from between the two rays at an angle π from the incident ray; see Figure 6.

Window Intersections Window intersections, as were used to eliminate nonshortest path windows [4,23], should not be used as they would eliminate valid geodesics.

Backtracking Care must be taken when backtracking (following a window back to the source) over vertices since the simple rule of selecting the shortest path back to the source no longer applies and since overlapping windows potentially give rise to multiple back-path choices.

Instead, any back-path between the two rays at an angle π from the incident back-path are potentially valid. Either application-dependent rules or back-path tracing can to be used to choose between more than one valid back-path.

Boundary Vertices In shortest-path algorithms, *boundary vertices* give rise to new pseudo-sources to allow shortest paths to curve around edges of the mesh. During our butterfly window propagation, boundary vertices should be ignored since candidate curves must be geodesic.

On the other hand, during source curve propagation, boundary vertices require special processing. As we do not propagate circular distance windows from the source curve's end points as do Bommes and Kobbelt [4] (our source curve effectively has no end points) our computed upper bound may not always reach the boundary of the mesh. This happens when the mesh boundary proceeds "outward" from the end of the source curve.

To resolve this problem, when an unreached boundary vertex is encountered, we unfold the boundary faces into the plane searching for the first point along the boundary that is δ away from the line supporting the source geodesic. Then we simply attach this boundary point to the end of our foreshortened upper bound.

Window Sorting As mentioned in Section 3.4, windows do not need to be sorted and propagated in path length order since we are not looking for the shortest path. Instead, windows need to be kept in sequential order to facilitate construction of a connected upper bound and propagation of areas from window to window.

Triangle Decimation Rather than an interpolated estimate combined with decimation as used in Bommes and Kobbelt [4], we need to find an exact iso-distance curve when forming our upper bound. Fortunately, candidates cannot contact these arc sections of this iso-distance curve (that always curve *back toward* the source) without immediately crossing the upper bound and violating the upper bound constraint (see Section 3.2 and Figure 7) — removing them from our consideration. We replace these arcs with line segments without loss of generality, simplifying the construction of our upper bound curve.

Aside from these differences, windows are propagated as described in the previous works mentioned.

5 Results

We now present results produced by our algorithm. First, we examine different factors that influence such results (Figure 10): the width bound δ, the initial curve orientation $\mathbf{v}$, and surface undulation.

Surface undulation appears to have the greatest effect on the strip tiling results, as shown in Figures 10(1-a) and 10(1-b). Our experiments show that while p, $\mathbf{v}$ and δ are held constant, increasing the surface undulation gives rise to a relatively abrupt transition from unsevered tilings as in 10(1-a) to highly severed tilings as in 10(1-b). Note that while the small triangles in 10(1-b) appear large, the width constraint is in fact satisfied as every point on the neighboring geodesic is within a geodesic distance δ from a point on its source.

Initial orientation With the surface and the width bound δ held constant, we observe that tilings are also relatively sensitive to the initial curve orientation $\mathbf{v}$. As shown in Figures 10(2-a) and 10(2-b), when the same geodesics are forced to travel through multiple regions of extreme curvature, they are more subject to severing than when these extremes are "spread out" amongst the geodesics.

It is also interesting to note that even though the altered orientation is noticeable in the top middle portions of 10(2-a) and 10(2-b), the severing process of the algorithm in 10(2-b) in effect *reverts the orientation back* toward the more favorable one in 10(2-a). It is not clear whether this is to be expected or is merely a coincidence as finding a favorable orientation was not a goal we hoped to achieve in this version of our algorithm.

Material width As Figures 10(3-a) and 10(3-b) show, once a tiling without severing is found, reducing δ does not necessarily induce severing as might be expected. Intuitively, this independence may be due to the expected existence of a continuum of nonintersecting geodesics between any two "parallel" geodesics (nonintersecting geodesics no more than δ geodesic distance away from

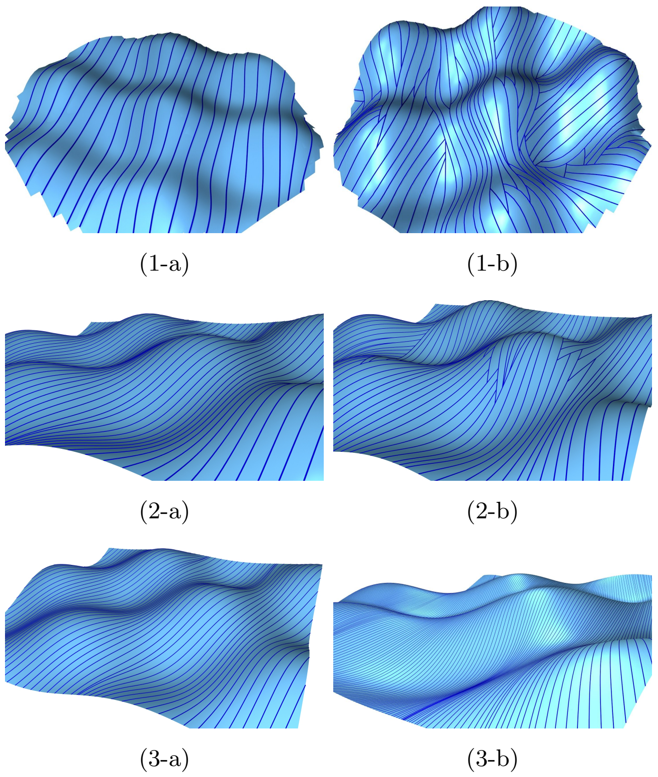


Fig. 10 Effects of different factors on straight strip tiling for a synthetic surface mimicking the roof of the Southern Cross Station (Figure 2). Row 1: Increased surface undulation in (1-b), holding **v** and δ fixed, causes the algorithm to sever tile lines (Section 3.5). Row 2: On the same surface and holding δ constant, a change in the curve orientation **v** again causes severing (2-b) as the geodesics are forced to travel through multiple regions of high curvature. Row 3: On the same surface while holding **v** unchanged, reducing the width bound δ in (3-b) does not introduce severing.

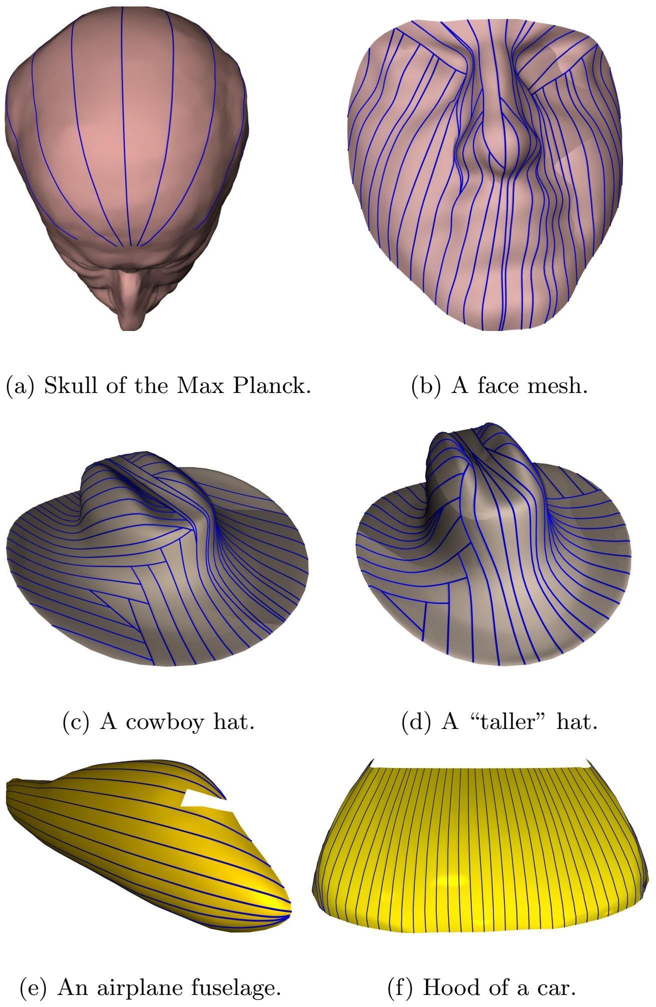


Fig. 11 A gallery of straight strip tiling results. See Table 1 for run times and material usage.

each other) on a sufficiently smooth surface. In fact, if the two geodesics belong to the same geodesic flow, then this is exactly the case. Exploring the applicability of geodesic flows to the tiling problem is left for future work.

Additional results, timing, material usage A set of additional results is shown in Figure 11 with the corresponding timing and material usage statistics provided in Table 1. Material usage is estimated as the surface area enclosed by neighboring geodesics divided by the area of the material strip, which is computed by the product of the longer length of the two neighboring geodesics and the width bound δ. When reporting timing, we do not vary **v** or δ as we recall from Section 3.4 that the algorithmic complexity depends only on the number of faces and number of (spherical) vertices in the input mesh. Figure 12 compares the observed run times to the expected computational complexity we claim in Section 3.4.

Mesh	Faces	Time	Material Usage
Skull	1,000	< 1 s	71 %
Hood	1,000	< 1 s	86 %
Hat	4,600	3 s	59 %
Taller hat	4,600	3 s	65 %
Face	6,100	4 s	66 %
Ship hull	7,800	5 s	79 %
Plane fuselage	9,700	15 s	81 %
Synthetic roof	19,600	20 s	75 %

Table 1 Numerical results of our experiments. All tests were run on a 2.60 GHz Pentium 4 with 2 GB of RAM. Timing results are reported in seconds.

Limitations Since the algorithm requires an initial, finite geodesic that spans the mesh, it does not work on closed surfaces. However, if a small hole is punched and a non-self-intersecting initial geodesic can be found that starts and ends at the hole boundary, then the algorithm should succeed under reasonable parameters. Also, since there is no provision for joining separated upper bound curves, the algorithm does not work on surfaces with interior holes. An effective means of handling both limitations is

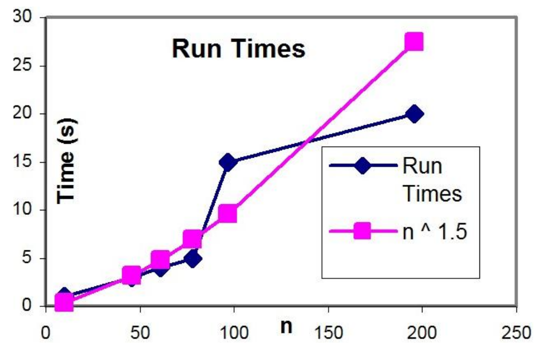


Fig. 12 Run times from Table 1 plotted against the claimed algorithmic complexity of $O(n^{1.5})$ from Section 3.4 (arbitrary constant chosen to fit curve to sample points). A reasonable correspondence exists with two outliers, the plane fuselage and the synthetic roof. Their deviation from the norm is due to the large difference in the proportion of spherical vertices on these models.

to first segment the given surface into open patches with simple boundaries before tiling.

Finally, we emphasize again that creating actual planar strips from the geodesics curves we obtain that can be joined to approximate an input surface with bounded error is a complex optimization problem which has been the subject of recent work [20]. The planar (yellow) strips shown in Figure 1(b) are merely meant to indicate the geodesic widths (over the input mesh) of the surface strips. While these plots primarily show that our chosen geodesics satisfy the prescribed width constraint, they are also intended to suggest how straight strips of material may be trimmed and joined to approximate the original surface. However it is important to note that the shapes in Figure 1(b) themselves generally cannot be joined together to approximate the surface.

6 Conclusion and future work

In this paper, we tackle the straight strip tiling problem on surfaces. Our approach relies on a width-bounded geodesic marching strategy, whereby starting from a user-specified initial point and orientation which define a source geodesic, subsequent geodesics are constructed in a greedy fashion one at a time to respect the width bound while attempting to minimize material waste. The algorithm proposed is only a preliminary attempt and it still leaves much room for improvement. In particular, although the geodesic construction component of the algorithm is exact with respect to the polyhedral geometry of the input mesh and the width bound is respected exactly as well, the material optimization part is far from rigorous — we have not provided a formal objective function.

The important question of when the algorithm would fail is also left unanswered. It is conceivable that certain curvature bounds may be found to address this issue. The question of whether there exists a "straightness-preserving" mesh parameterization scheme and the applicability of geodesic flows to the mesh setting to solve our tiling problem are both interesting as it is likely that a globally optimal technique, if found, would produce better results. Finally, we would like to remove the need for user initialization, seek an optimal general orientation for strip placements, and handle more complex boundary conditions arising from input surfaces with holes.

To the best of our knowledge, as we conduct the current research, the straight strip tiling problem is new. However during final preparation of the manuscript, we became aware of the recent, yet to be published, work of Pottmann et al. [19] which solves a similar problem. Relying on geodesic marching guided by the Jacobi field, they can obtain geodesic 1-patterns on a surface by starting with a source geodesic and iteratively computing neighboring geodesics approximately bounded by a material width measured geodesically from the source at sampled intervals. With their approach, constraints other than width bounds can be integrated. In contrast, our approach is designed specifically to satisfy the width bound producing guaranteed results with modest computational complexity.

Acknowledgements The authors would first like to thank the anonymous reviewers for their valuable feedback. Thanks should also go to Dr. Alla Sheffer and several members of the Graphics Usability and Visualization (GrUVi) lab at Simon Fraser University (SFU) — specifically Dr. Ramsay Dyer and Oliver van Kaick, for their discussions on the paper and the ideas therein. This work has been supported in part by an MITACS Internship grant entitled "Covering Surfaces with Strips".

References

1. Aleksandrov, A.D.: Intrinsic Geometry of Surfaces. American Mathematical Society (1967)
2. Alliez, P., Ucelli, G., Gotsman, C., Attene, M.: Recent advances in remeshing of surfaces. Tech. rep., AIM@SHAPE Network of Excellence (2005)
3. Berger, M.: A panoramic view of Riemannian geometry. Springer (2000)
4. Bommes, D., Kobbelt, L.: Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In: Proc. of Vision, Modeling, and Visualization (VMV), pp. 151–160 (2007)
5. Carmo, M.P.D.: Differential Geometry of Curves and Surfaces. Prentice-Hall (1976)
6. Dijkstra, E.: A note on two problems in connection with graphs. Numerische Mathematik **1**, 269–271 (1959)
7. Floater, M.S., Hormann, K.: Surface parameterization: a tutorial and survey. Advances in Multiresolution for Geometric Modelling pp. 157–186 (2005)
8. Grundig, L., Ekert, L., E., M.: Geodesic and semi-geodesic line algorithms for cutting pattern generation of architectural textile structures. Proc. of Asia-Pacific Conference on Shell and Spatial Structures (1996)
9. Grundig, L., Singer, P., Strbel, D., E., M.: High-performance cutting pattern generation of architectural textile structures. Proc. of International Colloquium on Computation of Shell And Spatial Structures (2000)

10. Kimmel, R., Sethian, J.: Computing geodesic paths on manifolds (1998)
11. Liu, Y., Pottmann, H., Wallner, J., Yang, Y.L., Wang, W.: Geometric modeling with conical meshes and developable surfaces. ACM Trans. Graph. **25**(3), 681–689 (2006)
12. Liu, Y.J., Zhou, Q.Y., Hu, S.M.: Handling degenerate cases in exact geodesic computation on triangle meshes. The Visusal Computer **23**(9), 661–668 (2007)
13. Mitchell, J.S.B., Mount, D.M., Papadimitriou, C.H.: The discrete geodesic problem. SIAM J. Comput. **16**(4), 647–668 (1987)
14. Paternain, G.P.: Geodesic Flows. Birkhauser (1999)
15. Pogorelov, A.V.: Quasi-geodesic lines on a convex surface. Mat. Sb. (N.S.) **25**(67), 275–306 (1949)
16. Polthier, K., Schmies, M.: Geodesic flow on polyhedral surfaces. In: Proc. of Eurographics-IEEE Symposium on Scientific Visualization, pp. 179–188 (1999)
17. Polthier, K., Schmies, M.: Straightest geodesics on polyhedral surfaces. In: ACM SIGGRAPH 2006 Courses, pp. 30–38 (2006)
18. Pottmann, H., Asperl, A., Hofer, M., Kilian, A.: Architectural Geometry. Bentley Institute Press (2007)
19. Pottmann, H., Huang, Q., Deng, B., Schiftner, A., Kilian, M., Guibas, L., Wallner, J.: Geodesic patterns. ACM Trans. Graphics **29**(3) (2010). URL http://www.geometrie.tugraz.at/wallner/geopattern.pdf. To appear
20. Pottmann, H., Schiftner, A., Bo, P., Schmiedhofer, H., Wang, W., Baldassini, N., Wallner, J.: Freeform surfaces from single curved panels. ACM Trans. Graphics **27**(3) (2008)
21. Pottmann, H., Schiftner, A., Wallner, J.: Geometry of architectural freeform structures. International Mathematical News **209**, 15–28 (2008)
22. Sheffer, A., Praun, E., Rose, K.: Mesh parameterization methods and their applications. Found. Trends. Comput. Graph. Vis. **2**(2), 105–171 (2006)
23. Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J., Hoppe, H.: Fast exact and approximate geodesics on meshes. ACM Trans. Graph. **24**(3), 553–560 (2005)
24. Vanček, P., Kolingerová, I.: Comparison of triangle strips algorithms. Computers and Graphics **31**(1), 100–118 (2007)

E. Joseph Kahlert is a Master's of Science graduate from the Graphics, Usability, and Visualization Lab in the School of Computing Science at Simon Fraser University, Canada. He also holds an M.B.A. from Nova Southeastern University as well as a B.A.Sc. from the University of British Columbia.

Matt Olson is a Ph.D. candidate in the Graphics, Usability, and Visualization lab at Simon Fraser University in Burnaby, Canada. He also holds a B.Sc. from the University of Alberta. His research interests include geometry processing and real-time rendering.

Hao Zhang co-directs the Graphics, Usability, and Visualization Lab at Simon Fraser University, Canada, where he is an associate professor in the School of Computing Science. He received his Ph.D. from the Dynamic Graphics Project (DGP), Department of Computer Science, University of Toronto in 2003 and M.Math. and B.Math. degrees from the University of Waterloo. His research interests include geometry processing and computer graphics. Recently, he has served on the program committees of Eurographics, SGP, Pacific Graphics, among others. He was a winner of the Best Paper Award from SGP 2008.