

GENERATING COMPRESSIBLE TRIANGULAR MESH

Laxmi P Gewali and C Sun
Department of Computer Science
University of Nevada, Las Vegas
Las Vegas NV 89014
{laxmi, schj}@cs.unlv.edu

Abstract

Compressible meshes are highly desired in application areas such as, interactive visualization, image rendering, and transmission of mesh data across the Internet. We present a critical review of recently reported algorithms dealing with the geometric compression of triangular mesh. We propose an algorithm based on the strip/core decomposition of polygons that yield compressible triangular mesh containing high proportion of quality triangles. We report an implementation of the spiral decomposition algorithm. The experimental results show that the proposed algorithm produces high quality mesh admitting significant compression.

Key Words: Compression, Computer Vision, Rendering, Visualization, Mesh Generation

1. Introduction

Triangular meshes are widely used to represent the surface of geometric objects. Since all topological surfaces can be partitioned into a mesh of triangular patches, triangular meshes are highly desirable in application areas such as rendering, interactive visualization, finite element analysis, and interpolation of scattered data. If the surface gradients of a geometric object are steep then exceedingly large number of triangles are needed to represent the surface. When such meshes are fed to graphics display engines, the rendering time critically depends on the rate at which the triangles are transmitted to the engine. To speedup the rendering task it is highly desirable to have a compact representation of the triangular mesh. A straightforward way of encoding a triangular mesh is to explicitly list all the triangles of the mesh in certain order. Such a representation is called a *complete enumeration*. In a complete enumeration scheme, $3*k$ vertices are needed to encode a mesh with k triangles, and many vertices are listed more than once.

Geometric compression is a way of representing a mesh in a compact form so that the multiple-occurrence of vertices of the mesh is reduced. Ideally, we would like to have exactly one representation of each vertex.

However, it is very easy to find that the ideal representation is possible only for a very restricted class of triangular meshes. Geometric compression is achieved by taking advantage of spatial coherence of triangles in the mesh. The idea is to encode the mesh by listing a sequence of triangles where adjacent triangles in the sequence share an edge. Such a sequence is referred to as a *triangular strip*. If a mesh admits a triangular strip representation then the vertices shared by adjacent triangles can be compactly coded without the need of explicit enumeration. Most of the geometric compression algorithms reported in the literature are directly or indirectly based on the use of the properties of adjacent triangles in the mesh.

Algorithms for geometric compression have been mainly considered in two areas of computer science: (i) computer graphics and (ii) computational geometry. In computer graphics community, methods including (a) *back reference stack* [1], (b) *vertex/edge split*, (c) *hierarchical decomposition* [2], and (d) *topological surgery* [3]. In computational geometry community, the geometric compression problem is viewed as the problem for finding Hamiltonian path in the dual graph of the mesh. Some of the compression algorithms based on this view include (a) *spanning tree method* [4], (b) *patchification* method [5], and (c) *search tree method* [6]. Although several algorithms have been reported dealing with the compression of triangular mesh, no work is known that addresses both the compressibility and the quality of the generated mesh. A good quality mesh should have high proportion of 'fat' triangles. (A triangle is called *fat* if the lengths of its sides do not differ much.) It may be noted that in most numerical applications of triangular mesh, including the interpolation of scattered data, the accuracy of computed result improves with the increase in the quality of the underlying mesh.

In this paper we address the problem of generating compressible triangular mesh that contain high proportion of quality triangles. Section 2 deals with the background, preliminaries, and previous work related to geometric compression. In Section 3 we propose an algorithm based on strip/core decomposition that generates high quality triangular mesh admitting significant compression in convex domain. We then consider how to apply the spiral decomposition approach for generating high quality compressible

triangular mesh in non-convex polygons. Finally, in Section 4 we report the experimental results produced by an implementation of the proposed strip/core

2. Preliminaries and Previous Work

Consider the partitioning of a simple polygon into a triangular mesh (Figure 1). A straightforward way of encoding a mesh consisting of k triangles is to represent all triangles by listing three vertices per triangle. Such a listing is referred to as **complete listing**. One of the encoding of the mesh in Figure 1a, based on complete listing is:

[t8 t7 t2 t6 t3 t1 t5 t4]
 =
 [abj jbc jcd jid idf fde hfg ghf]

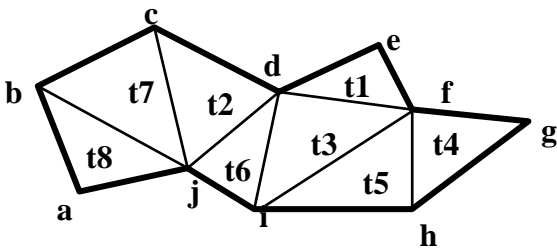


Figure 1a

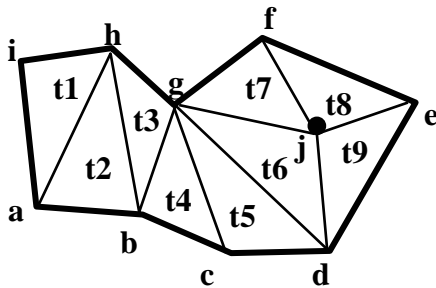


Figure 1b

The size of the encoding of a mesh is measured in term of

the number of bits needed to represent the encoding. It is clear that the size of the encoding of a mesh with k triangles, based on complete listing is $3*k*s$, where s is the number of bits needed to encode a coordinate. An ordered sequence of distinct triangles $t_1, t_2, t_3, \dots, t_m$ is called a **triangular strip** if all adjacent triangles in the sequence share an edge. For example, the sequence $t_1, t_2, t_3, \dots, t_9$ in Figure 1b is a triangular strip. Readers can easily verify that the mesh of Figure 1a can not be listed as a triangle strip. If a mesh can be represented by a triangular-strip then one can encode the mesh in a compact form by making the use of shared edges of adjacent triangles in the sequence. If the strip has k triangles then we can encode it by listing $k+2$ vertices (coordinates) and $k-1$ bits. The encoding sequence is formed by arranging vertices and bits alternately, except for the first two vertices. Let v_i and b_i denote the i th

decomposition algorithm and discuss its extensions and generalization.

vertex and i th bit in the sequence. Then the encoding sequence has the form:

$$v_1 v_2 v_3 b_1 v_4 b_2 v_5 b_3 \dots b_{k-2} v_{k-1} b_{k-1} v_k$$

In this representation, if $v_i v_{i+1} v_{i+2}$ form a triangle then bit occurring after v_{i+2} is b_i . The next triangle in the sequence is formed either by $v_i v_{i+2} v_{i+3}$ or by $v_i v_{i+1} v_{i+3}$. Bit b_i is set 1 if the next triangle is given by $v_i v_{i+2} v_{i+3}$; otherwise (next triangle is given by $v_i v_{i+1} v_{i+3}$) b_i is set 0. For example, the triangle strip of Figure 1b can be written as:

$$a i h 0 b 0 g 0 c 0 d 1 j 1 f 0 0 1 d$$

From this sequence the actual triangle can be decoded as:

$$a i h a h b h b g b g c g c d g d j g j f g f e j e d$$

Observation 1: If a mesh with k triangles can be encoded by a triangle strip then the size of the encoding is given by $(k-2)s+(k-1)$ bits, where s is the number of bits required to represent a coordinate

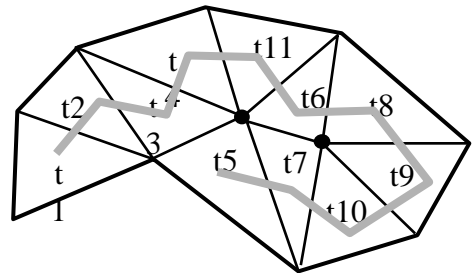


Figure 2a: Illustrating Hamiltonian Mesh

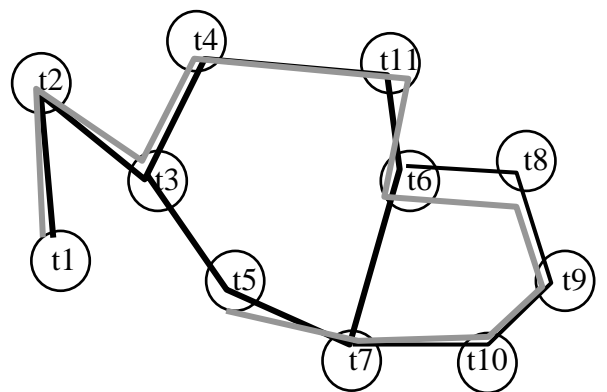


Figure 2b: Hamiltonian Path in the Dual Graph

A triangular mesh is called Hamiltonian if its dual graph contains a Hamiltonian cycle/path. Finding a Hamiltonian path in a graph is known to be intractable [7]. But a triangular mesh is a restricted version of the planar graph and hence the problem of finding a triangle strip in a triangular mesh could be potentially a tractable problem. However, it was recently established

that finding a Hamiltonian triangulation in a polygon with holes is NP-hard [4].

Several algorithms are available in the literature for generating a triangular mesh in polygonal domain. Constraint Delaunay triangulation, advancing wave front, are some of the widely used techniques. Meshes generated by most of these algorithms are not necessarily Hamiltonian. It would be very useful to develop algorithms that generate triangular meshes having Hamiltonian property.

Spanning Tree Approach

Any triangular mesh can be made Hamiltonian by refining each triangle of the mesh and by adding/removing few edges. This approach suggested in [4] uses a spanning tree of the original mesh as the skeleton of the desired Hamiltonian path. The generated triangle strip is essentially the Euler tour of the spanning tree.

A serious drawback of the spanning tree approach for generating Hamiltonian mesh is the quality of the triangles in the mesh. The triangles at the boundary of the polygon are thin and long. This is due to the fact that many of the new triangles' interior angles are much smaller than the interior angles of the original triangles. Hence for applications requiring good quality Hamiltonian mesh the spanning tree approach is not suitable.

Hierarchical Decomposition:

Recently, an approach based on hierarchical decomposition of polygons for generating a Hamiltonian triangular mesh is reported in [2]. The algorithm starts from a very coarse Hamiltonian triangular mesh. (The assumption of starting from a very coarse Hamiltonian mesh is a realistic assumption. Any polygon can be partitioned into four parts whose dual graph admits a Hamiltonian cycle.) Elements of the coarse mesh are successively refined to make sure that the refined mesh still admits Hamiltonian cycle. It is established in [2] that one can always refine the mesh by preserving the Hamiltonian property when a triangle is refined by using appropriate templates. This approach generates a good quality mesh only when the elements of the coarse mesh are close to isosceles right-angled triangles. If we apply hierarchical decomposition approach to a polygonal domain with a small internal angle(s) then the quality of the generated Hamiltonian mesh deteriorates.

3. Strip-Core Decomposition

In this section we propose an algorithm that generates a good quality Hamiltonian mesh for simple polygons. Our algorithm is based on the idea of first partitioning the polygonal domain into convex components and then apply the technique of spiral decomposition to generate Hamiltonian mesh on each convex components. We then fuse the Hamiltonian mesh on each component to generate a Hamiltonian mesh for the entire polygonal

domain. We first describe the algorithm for convex polygons. Given a convex polygon and a positive number δ representing the approximate size of the mesh element, we want to design an algorithm that constructs a Hamiltonian mesh of the polygon by adding steiner vertices. We also want the length of the sides of each triangle not much longer or shorter than δ . The second requirement ensures the quality of the generated mesh. One way to measure the quality of a triangle is to compute the maximum of the absolute values of the deviations of the lengths of the sides of the triangle from δ . More precisely, if a, b, c are the lengths of the sides of a triangle then its quality Q is given by $Q = \max(|a-\delta|, |b-\delta|, |c-\delta|)$. Another way of measuring the quality of a triangle is in terms of the ratio of the difference of the lengths of its shortest and longest sides to the length of its longest side. Under this scheme the quality is best if it is closer to zero. Of course, the quality of the whole mesh increases if it contains a high proportion of quality triangles.

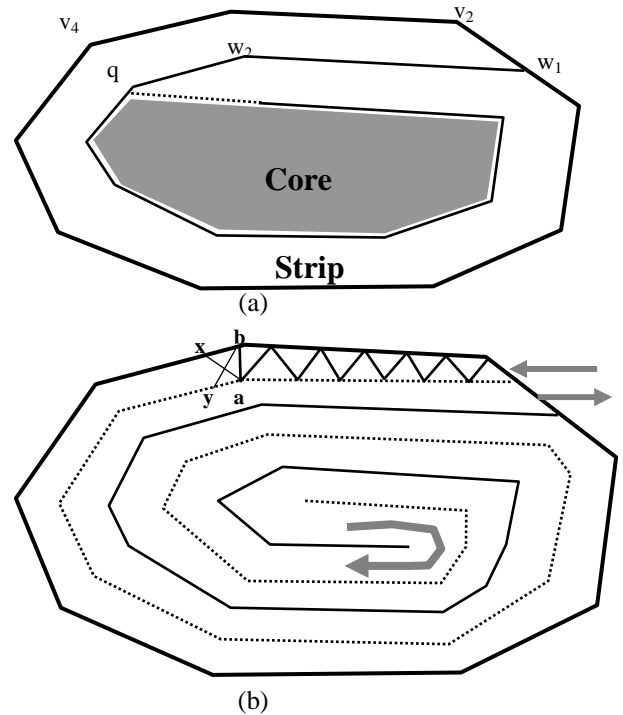


Figure 3: Illustrating Strip/Core Decomposition

The algorithm partitions each convex component into two *spiral-strips*: one spiraling outside-in and the other spiraling inside-out. Each spiral-strip is triangulated to obtain the Hamiltonian triangulation of the convex component. The width of the generated spirals can be made fairly uniform and close to the given value δ . This leads to a triangulation with a high proportion of better quality triangles. We describe our algorithm with a running example. Consider a convex polygon whose minor diameter is much larger than the size factor δ (Figure 3). Let the vertices of the polygon be $v_1, v_2, v_3, \dots, v_n$. The boundary of the polygon is partitioned into edges close to δ by introducing steiner vertices. We next choose a

vertex (say w_j) about 2δ away from a non-steiner vertex (say v_j) and start to build the inward spiraling convex chain. For example, the first segment

(w_b, w_2) of the inward chain is constructed by starting with a segment parallel to (v_2, v_3) . This makes the width of the spiral strip close to 2δ . If the length of the constructed segment is much smaller than δ then we choose the direction given by (v_3, v_4) as the guiding direction to construct the segment(w_b, w_2). We construct the next segment of the inward spiral by following the direction of the guiding segment determined by the next segment of the last guiding segment. When we proceed in this manner we come back to the original starting point w_j . From that point onward we pick the guiding direction by following the previous spiraling layer, i.e., along w_j, w_2, w_3 etc.. With the partially constructed spiraling chain, we can define a smaller convex polygon called the **core of the polygon** as defined below.

Core of a convex polygon: Extend the last segment (i.e., the innermost segment of the spiraling chain) so that the extension meets the chain at a point, say, q . (The extension is shown by dashed segment in Figure 3a.) The inner most convex polygon formed in this way is the **core** of the polygon with respect to the spiraling chain. Each time a new segment of the spiral is constructed we check for the minor diameter of the core. The construction of the inward spiral stops if the minor diameter of the core is smaller than 3δ . When the construction of the inward spiral is completed, the outward spiral is constructed by following the middle points of the inward spiraling strip. Outward spiral chain is shown by dotted edges in Figure 3b. The resulting structure is the **strip/core decomposition** of the convex polygon. The decomposition of a convex polygon into spiral-strip and core can be implemented by performing the process of shrinking a convex chain.

An **open convex-chain** is a connected proper subset of the boundary of a convex polygon. A convex chain c is called **completely-open** if c can be contained within the tangential parallel lines passing through the end points of c (Figure 4a).

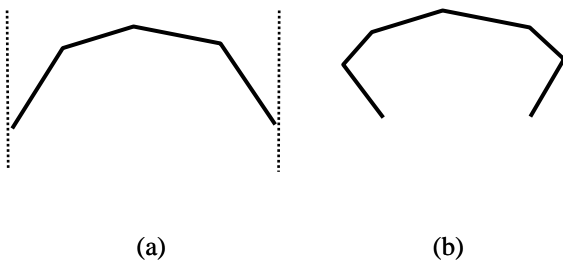


Figure 4: Two Types of Convex Chains

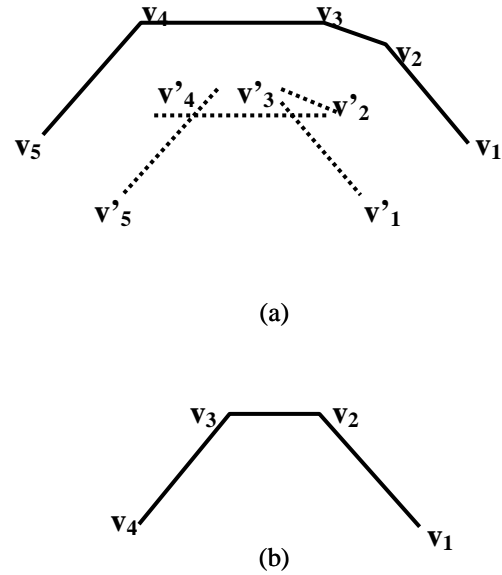


Figure 5: Illustrating the Shrinking of a Convex Chain

Consider a completely open convex chain (v_1, v_2, \dots, v_n) (drawn by solid edges in Figure 5a). Corresponding to each segment (v_b, v_{i+1}) of the chain, construct a shifted segment (v'_b, v'_{i+1}) obtained by shifting (v_b, v_{i+1}) by distance close to 2δ toward the interior of the chain. The shifted segments are drawn dashed in Figure 5a. The convex chain obtained by connecting the end points and intersection points of the corresponding shifted segment give the **shrunk chain**. The shrunk chain is shown in Figure 5b. It is observed that the shrunk chain may contain less number of segments than in the original chain. Some segments are lost in the shrinking process. Given a convex chain and a distance 2δ , we can construct the shrunk chain in linear time ($O(n)$) as follows. To construct the shrunk chain, we examine the line segments of the original chain one by one starting from the segment at one end of the original chain. The line (v'_1, v'_2) is taken as the initial shrunk chain, where (v'_1, v'_2) is the line segment parallel to segment (v_1, v_2) and distance 2δ toward the interior of the original chain. At the i^{th} Step, i segments are included in the shrunk chain and k_i segments in the original chain have been already processed. To include the effect of the next segment ($k+1^{th}$ segment) of original chain to the partially constructed shrunk chain, segment (v_k, v_{k+1}) is shifted parallel to form ray, $s_k = (v'_k, v'_{k+1})$, starting at v'_k and extending toward v'_{k+1} . We then find the intersection point v'_x between s_k and the partially constructed shrunk chain. The updated shrunk-chain is formed by removing a portion of the currently formed shrunk-chain lying on the appropriate side of s_k (Figure 5).

This construction is repeated until all segments of the original chain are processed. The above description is sketched formally in *Algorithm Shrunk-Chain*.

Algorithm Shrunk-Chain

Input: A convex chain ch , integer δ .

Output: A convex chain ch' formed by shrinking ch by 2δ .

Step 1a: Construct segment $s'_1 = (v'_1, v'_2)$ parallel to segment (v_1, v_2) and distance 2δ from segment (v_1, v_2) .

Step 1b: Chain $ch' =$ line l'_i passing through segment s'_1 .

Step 1c: $i = 2$.

Step 2: While all segments of ch are not processed do

Step 3: Construct segment s'_k parallel to (v_i, v_{i+1}) and distance δ from segment (v_i, v_{i+1}) .

Step 4: Construct a ray r'_i starting at v'_i and extending toward v'_{i+1} .

Step 5: Find the intersection point v'_x between ch' and r'_i .

Step 6: Let ch'' be the portion of ch' that lie to the side of s'_k containing v'_1 . Replace ch' by ch'' . Set i to $i+1$.

Lemma 1: The execution time of the algorithm Shrunk-Chain is $O(n)$, where n is the number of segments in the original chain.

Proof (Omitted)

Triangulating Spiral Strips

The boundaries of spiral strips are partitioned into segments of length close to δ . The spiral strip is partitioned into triangles by adding diagonals. Consider a partially triangulated spiral strip as shown in Figure 3b. Let (a, b) be the most recently inserted diagonal in the spiral strip. Let x and y be the next vertices of b and a in the respective chains. The next diagonal to insert is either (x, a) or (y, b) .

With respect to a candidate diagonal, say (x, a) , we examine five angles - three internal angles and two external angles. The smallest angle among these five angles is referred to as the **limit angle** of the candidate diagonal. Similarly, the limit angle of the other diagonal (y, b) is determined. The diagonal that maximizes the limit angle is used to form the next triangle. Triangles constructed in this way increase the quality of the mesh.

Since the minor diameter of the core close to 3δ the core can be triangulated by using templates so that it can be attached to the triangulated strips to make the entire triangulation Hamiltonian.

The time needed to produce the compressible triangular mesh can be expressed in term of the total number of nodes m in the mesh and the total number of nodes n on the boundary of the input polygonal domain. Each segment of the spiral chain is constructed in constant time by following the boundary of the previous layer. When a segment is constructed the algorithm checks for the size of the core. $O(n)$ time is needed to check the size of the core. Thus the total time needed to perform

spiral decomposition is $O(n*m)$. Once spiral decomposition is available, triangulation can be done at the cost of constant time per inserted diagonal. Hence the total time for the whole algorithm is $O(n*m)$.

Theorem 1: Given a convex polygon P and size factor δ , a Hamiltonian triangular mesh for P can be obtained in $O(m*n)$ time, where n is the number of vertices in P and m is the number of vertices in the generated mesh.

Hamiltonian Mesh for Non-Convex Polygons

The strip/core decomposition algorithm presented above can not be directly applied if the polygon is not convex. If the input polygon is non-convex then we first partition the polygon into convex components and apply the strip/core decomposition approach to each component. The detail of fusing Hamiltonian cycles is omitted in this paper due to lack of space. The fusing step is illustrated in Figure 6.

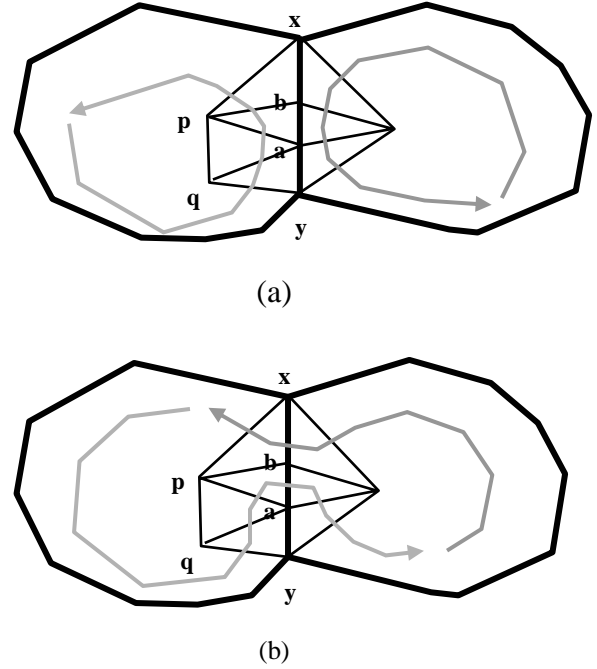


Figure 6: Illustrating the Fusing of the Hamiltonian Paths

4. Experimental Results and Discussion

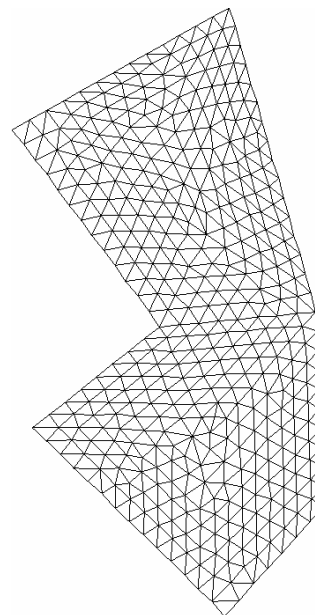
We implemented the proposed strip-core decomposition algorithm in JAVA programming language. We used doubly connected edge list (dcel) data structure to represent the generated mesh. An example of generated mesh is shown in Figure 7, where the first part of the figure shows the generated mesh and the second part shows the Hamiltonian cycle drawn over the mesh. As expected triangles in the regions corresponding to strips are of better quality compared to the triangles in the region near the core of the convex components. We computed quality factors for several

input polygons and in all test examples 90% of the triangles had quality factor within 0.5.

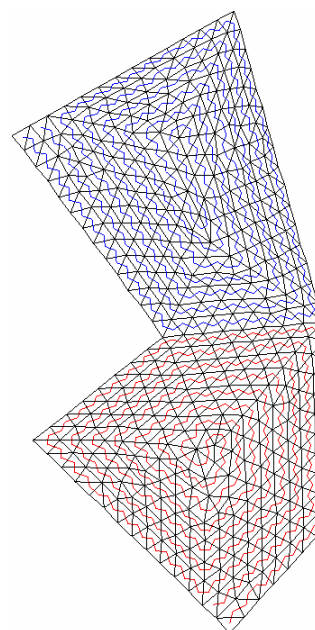
The observed experimental results suggest that the approach based on strip/core decomposition generates good quality triangular Hamiltonian mesh. The qualities of the generated meshes are better for those convex components whose aspect ratio is close to 1. This suggests that if we can partition the polygonal domain into convex components having aspect ratios not far from 1 then the overall quality of the generated Hamiltonian mesh should improve significantly

References

- [1] M. Dering, "Geometry Compression," *Computer Graphics Proceedings – ACM SIGGRAPH*, pp. 13-22, 1995.
- [2] L. Velho, L. D. Defigueiredo, and J. Gomez, "Hierarchical Generalized Triangle Strip," *The Visual Computer*, 15, pp. 21-35, 1999.
- [3] G. Taubin and J. Rossignac, "Geometric Compression Through Topological Surgery," *ACM Trans. Graph.*, 17: (2), pp. 84-115, 1998.
- [4] E. M. Arkin, M. Held, J. S. B. Mitchell, and S. Skiena, "Hamiltonian Triangulation for Fast Rendering," *The Visual Computer*, 12, pp. 429-444, 1996.
- [5] F. Evans, S. Skiena, and Amitabh Varshney, "Efficiently Generating Triangle Strips for Fast Rendering," Manuscript, Department of Computer Science, State University of New York, Stony Brook, 1999.
- [6] X. Xiang, M. Held, and J. S. B. Mitchell, "Fast and Effective Stripification of Polygonal Surface Models," Manuscript, Dept. of Applied Math and Statistics, State Univ of New York, Stony Brook, 1999.
- [7] M. R. Garey and D. S. Johnson, "Computers and Intractability," W. H. Freeman, 1979.



(a): Generated Mesh



(b): Hamiltonian Path on the Mesh

Figure 7: The Mesh Generated by the Proposed Algorithm