EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (*2005*), *Number 3*

Hierarchyless Simplification, Stripification and Compression of Triangulated Two-Manifolds

Pablo Diaz-Gutierrez M. Gopi Renato Pajarola
Department of Computer Science,
University of California, Irvine

Abstract

In this paper we explore the algorithmic space in which stripification, simplification and geometric compression of triangulated 2-manifolds overlap. Edge-collapse/uncollapse based geometric simplification algorithms develop a hierarchy of collapses such that during uncollapse the reverse order has to be maintained. We show that restricting the simplification and refinement operations only to, what we call, the collapsible edges *creates* hierarchyless simplification *in which the operations on one edge can be performed independent of those on another. Although only a restricted set of edges is used for simplification operations, we prove topological results to show that, with minor retriangulation, any triangulated 2-manifold can be reduced to either a single vertex or a single edge using the hierarchyless simplification, resulting in extreme simplification.*
The set of collapsible edges helps us analyze and relate the similarities between simplification, stripification and geometric compression algorithms. We show that the maximal set of collapsible edges implicitly describes a triangle strip representation of the original model. Further, these strips can be effortlessly maintained on multi-resolution models obtained through any sequence of hierarchyless simplifications on these collapsible edges. Due to natural relationship between stripification and geometric compression, these multi-resolution models can also be efficiently compressed using traditional compression algorithms.
We present algorithms to find the maximal set of collapsible edges and to reorganize these edges to get the minimum number of connected components of these edges. An order-independent simplification and refinement of these edges is achieved by our novel data structure and we show the results of our implementation of view-dependent, dynamic, hierarchyless simplification. We maintain a single triangle strip across all multi-resolution models created by the view-dependent simplification process. We present a new algorithm to compress the models using the triangle strips implicitly defined by the collapsible edges.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Geometric algorithms, Triangulation, Stripification, Compression, Simplification G.2.2 [Graph algorithms]: Hamiltonian Cycle, Perfect Matching.

1. Introduction

A popular approach for geometric simplification of triangulated models has been through edge-collapse and uncollapse (vertex-split) operations [HDD*93]. Using these basic operations, researchers have considered various aspects of the problem including different cost functions for high quality simplification [Hop96, GH97, Lue01], memory and speed efficient data structures [LT98, Paj01], dynamic view-dependent simplification [XV96, Hop97, ESV99], out-of-core simplification [ESC00, DP02], selectively refining a simplified model [KL03], and even multi-cost functions optimization for attribute preserving simplification [COM98, GH98]. Similarly, there is a reasonable amount of literature for triangle strip generation (e.g. [ESV96, Kor99, XHM99, Gop04]) and geometric compression [TDG*00] of triangulated models. Even though there is a strong relationship between stripification and compression there is little explicit work that relates these two fields besides [Dee95, Cho97, Ise00]. Further, there are only limited number of works relating simplification and stripification; existing literature primarily discuss the issue of maintaining strips during dynamic simplification (i.e. [ESAV99, BRR*01, Ste01, SP03]).

In this paper, we explore the common algorithmic space

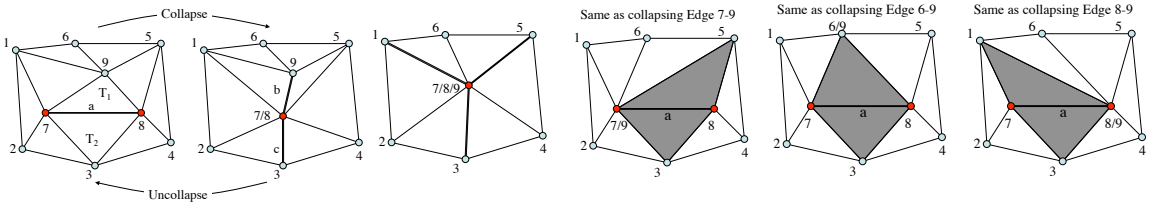

Figure 1: *Left to right: Conventional edge collapse and uncollapse operations. Multi-edges* b *and* c *are formed by collapsing edge* a*. Further collapse of* b *merges all the three edges of triangle* T_1*. Uncollapsing* a *without previously uncollapsing* b *results in ambiguity in the choice of edges to split: Three possibilities are shown, each of which is equivalent to a different single-edge collapse in the original model. Such an ambiguity appears when collapse of* n *number of edges results in the collapse of more than* n *edges to a single vertex. Traditionally, this ambiguity is handled by maintaining a hierarchy of collapses and following a strict reverse order of dependency during uncollapses. Hierarchyless simplification avoids this ambiguity by prohibiting two edges of the same triangle to be collapsed.*

between simplification, stripification, and geometric compression and lay a few theoretical foundations for the existence of such a space. Clearly, since we attempt to look into the *intersection* of these three fields, each of these fields will have a restricted scope. For example, we will not allow all edge-collapses but only those on chosen edges; not all possible strip configurations but only strip-loops; and finally, not all compression techniques but only those that produce triangle strips as a by-product during compression. Nevertheless, within this restricted space, we still can achieve extreme simplification of geometric models and reduce them to a single vertex or a single edge and can achieve a single strip-loop that covers the entire and every multi-resolution model in the process of simplification. Interestingly, by restricting the scope of each of these fields, we are expanding the scope of algorithmic applicability in each other field.

1.1. Main Contributions

Following are the main contributions of this paper.

- We introduce *hierarchyless simplification* in which the simplification operations on one edge is independent of the operations on other edges. This simplification is done on a select set of edges, what we call the *collapsible edges*. We prove the existence of and provide an algorithm to find the maximal number of *collapsible edges*.
- We prove topological properties of *hierarchyless simplification* that, with minor changes in the triangulation, any two manifold with arbitrary genus can be reduced to a single vertex or a single edge.
- We prove that the collapsible edges implicitly define a stripification of the model that is again implicitly maintained over all the multi-resolution models created by the simplification process.
- Finally, the triangle strip that represents the given model can efficiently be encoded by classical triangle mesh compression algorithms which use region growing or vertex-spanning tree based coding methods. Since the strip is defined by the *collapsible edges* which are the only edges used by the hierarchyless simplification, the compressed model can be viewed as the compression of the entire multi-resolution representation.

2. Hierarchyless Simplification

An edge collapse operation merges two adjacent vertices of the mesh. In Figure 1a, the shared edge *a* between triangles T_1 and T_2 is collapsed. The other two edges of each of the triangles merge to form new "multi-edges" *b* and *c* (Figure 1b). Traditional simplification algorithms allow further collapse of these "multi-edges" (Figure 1b). After collapsing the multi-edge *b*, if edge *a* has to be uncollapsed then it introduces an ambiguity, called the *split ambiguity*, as to which incident edges have to be split to introduce two new triangles. Even in a simple example of two edge collapses shown in Figure 1 we can see three possibilities just on one side of the uncollapsed edge *a*. It becomes more complicated with deeper nesting of collapses and uncollapses. Traditional methods *handle* this ambiguity by *restricting the order* of collapse and uncollapse operations; edge *b* is uncollapsed before edge *a*. This dependency between the operations creates a hierarchy of edges for dynamic simplification. In this paper, we introduce a method that *eliminates* this ambiguity by *restricting the edges* that are collapsed and uncollapsed.

It can be shown that the following four conditions are equivalent: (a) there exists a *split ambiguity*; (b) collapsing *n* edges has resulted in the collapse of more than *n* edges to a single vertex; (c) more than one edge of a triangle has been collapsed; (d) a multi-edge has been collapsed. Hence if one of these conditions is avoided then the *split ambiguity* is eliminated.

Definition 1 An *hierarchyless simplification* is an edge-collapse/uncollapse based geometric simplification process in which the multi-edges are not collapsible.

In a manifold, this condition enables any collapsed edge to

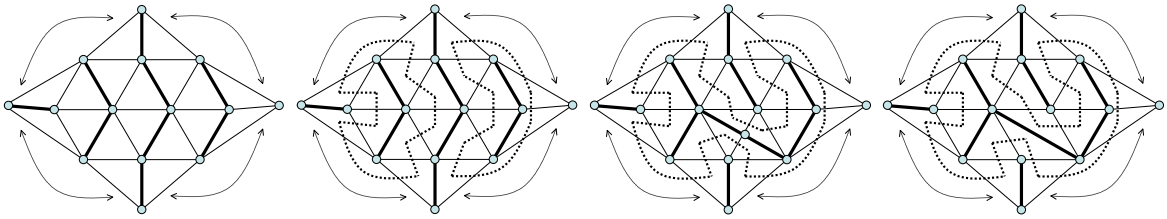

Figure 2: *From left to right: (a) The given triangulation of a manifold with genus 0. Assume the adjacency of boundary edges as shown by dotted arrows and all the boundary triangles share a common vertex. A maximal set of collapsible edges are shown as dark edges. Notice one and only one collapsible edge for every triangle. (b) Triangles are connected through non-collapsible edges to form two triangle strip loops. There are two connected collapsible-edge components forming interior medial axis of each of these loops and one forming the exterior medial axis of both these cycles. (c) Triangle-split operation: An adjacent triangle pair in two different cycles are split at the mid-point of the collapsible edge and the collapsible edge assignment is toggled around this vertex. This merges the two cycles and their interior medial axes collapsible edge components. (d) An alternative approach to triangle-split operation is an edge-swap operation which also connects the cycles and their interior medial axes.*

be uncollapsed in any arbitrary order and the connectivity in the neighborhood uniquely determined without any dependency on other collapsed edges. We prove this property in Appendix A. The data structure to effect such an order independency in operations and to efficiently find the local connectivity after every operation is detailed in Section 5.

Since we collapse only a restricted set of edges, the first topological question is to know the smallest model that can be got through hierarchyless simplification. Further, since the choice of an edge as collapsible prohibits a few other edges from being collapsible, it is important to know the best choice of edges so that we can maximize the number of collapsible edges to get the smallest model. We call a model *maximally face-simplifiable* if all triangles can be collapsed; in other words, since two triangles are removed per collapse, the model has $F/2$ collapsible edges, where F is the number of faces in the model. *Not all triangulated models are maximally face-simplifiable.* For example, models with boundaries are not guaranteed to be maximally face-simplifiable.

Theorem 1 Any triangulated two-manifold with arbitrary genus is maximally face-simplifiable.

Proof We define *partner triangles* to be the two triangles incident on a collapsible edge. Clearly, every triangle can have at most one partner triangle. The fundamental technique we use to arrive at maximum number of partner triangle pairs is a perfect matching algorithm. A *matching* in a graph $G = (V,E)$ is a subset M of the edges E such that no two edges in M share a common end node. A *perfect matching* M in G is a matching such that each node of G is incident to an edge in M. For a bridgeless graph in which every vertex has degree three, there always exists a perfect matching [Pet91]. In our case, we are interested in the dual graphs of triangulated two-manifolds; such graphs are bridgeless (in fact, 3-connected) and have degree three. A perfect matching in this dual graph will pair every triangle with exactly one of its adjacent triangles in the primal mesh. Hence any triangulated two-manifold with arbitrary genus is maximally face-simplifiable. Further, such a collapsible edge set, in other words a perfect matching in the dual graph can be found in time $O(n)$ for planar graphs [BBDL01] or $O(n\log^4 n)$ in general, where n is the number of input nodes in the dual graph; the latter bound can be further improved to $O(n\log^3 n\log\log n)$ using recent results of Thorup [Tho00]. A simple example of a triangulation of a genus zero object and a set of collapsible edges is shown in Figure 2a. □

Weighted Perfect Matching

High quality geometric simplification can be achieved only if the collapsible edges are chosen based on its cost of collapse. We construct the dual graph of the triangulation in which the cost of collapsing an edge in the mesh is assigned as the weight of the corresponding edge in the graph. We compute this cost as the accumulated quadric error of the resulting point after collapsing an edge. A publicly available weighted perfect matching algorithm on this graph will choose the matching edges such that the sum of the weights of the chosen edges is minimized.

3. Simplification and Stripification

There is a natural relationship between hierarchyless simplification of maximally face-simplifiable model and triangle-strip representation of that model. Specifically,

Lemma 1 Maximum set of collapsible edges in a triangulated two manifold implicitly defines a stripification of the entire model.

Proof Since a triangulated two manifold is maximally face-simplifiable, every triangle has one and only one collapsible edge and two non-collapsible edges. The sequences of triangles defined by the adjacency relationship across non-collapsible edges form a collection of disjoint triangle strip

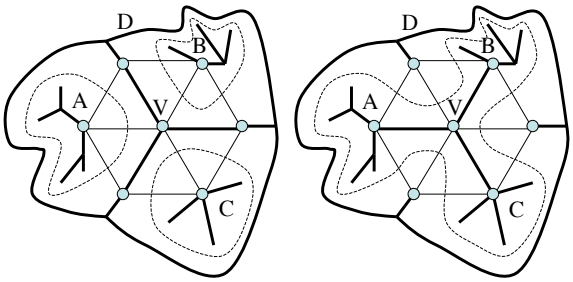

Figure 3: *An even-degree vertex with alternating collapsible edges is called a nodal-vertex. By toggling the collapsible and non-collapsible edges around this vertex, triangle strips and corresponding collapsible edge trees (A,B, and C) can be merged. It can be proved that the connected component of collapsible edges at the nodal vertex (D) cycles around each of the triangle strip loops (A, B, and C) incident on the nodal vertex. Hence the toggling operation around the nodal vertex does not disconnect this connected component (D).*

loops, the union of which covers all triangles of the mesh. An example of triangle strips defined by the collapsible edges is shown in Figure 2b. □

The stripification work by Gopi and Eppstein [GE04], in fact, exploits the perfect matching algorithm to find the triangle strips. The collapsible edges form the 'medial axes' of triangle strip loops (refer Figure 2b). We use this property to do *extreme hierarchyless simplification* as shown in the next section.

3.1. Extreme Simplification and Single Stripification

Though, in a maximally face-simplifiable model, all faces can be collapsed using hierarchyless simplification, the simplified model need not have minimum number of vertices. In order to minimize the number of vertices after all possible edge collapses, we have to reduce the number of connected components of collapsible edges, as each connected component collapses to one vertex. The number of connected components of collapsible edges is related to the number of triangle strip loops since the collapsible edges form *medial axes* of these strip loops.

Theorem 2 If s triangle strip loops cover the entire triangulated two manifold of genus g then the number of connected collapsible edge components $c > 0$ and $s+1-g \leq c \leq s+1$.

Proof of this theorem is given in Appendix B. Specifically, for a model with an arbitrary genus, if s is one then c is either one or two. In other words, *in extreme hierarchyless simplification, an oriented triangulated two manifold represented using a single strip loop can be simplified to either one vertex or two vertices (an edge) irrespective of its topology*.

Single triangle-strip representation of a manifold is achieved in [GE04] using two operations: nodal vertex processing and triangle-split. A nodal vertex in a mesh is one that has an even number e of triangles incident on it and these e triangles belong exactly to $e/2$ disjoint cycles (refer to Figure 3). The matched (collapsible) and unmatched (uncollapsible) edges alternate around such a vertex. In nodal vertex processing, the assignment of each of these edges is toggled to merge all the incident disjoint triangle strip loops.

Simplification operation is basically a weighted retriangulation of the mesh. The second operation to reduce the number of triangle strip loops, the *triangle-split*, retriangulates the model (Figure 2c). It splits two adjacent triangles belonging to two different strip loops into four triangles and performs a nodal vertex processing on the newly introduced vertex to merge these two loops. In addition to the above operation, we introduce a new operation that is equivalent to triangle-split, the edge-swap operation (Figure 2d), in which we swap a collapsible edge separating two different strip loops. A triangle split operation retains the visual appearance of the geometry by introducing the new vertex at the midpoint of the common edge. An edge swap operation does not add new triangles but might change the visual appearance due to change in geometry. We propose to use the edge-swap operation in the planar regions of the mesh (in which visual appearance is least affected by swapping edges), and to apply only the triangle-split operation in all other regions.

Even though adding more triangles to later simplify them seems contradicting, merging connected components through this process gives us incredible flexibility in efficiently managing the data during run-time simplification. The added edges with potentially higher costs for simplification might in any case be collapsed only during extreme simplification. Hence the quality of simplification is not affected in the early stages of simplification. More importantly, even though the theoretical upper bound is 50% [GE04], empirically we add less than only 3% more triangles to the model. Hence this addition has negligible effect in the overall quality of the simplified model but improves the run-time efficiency and later, topological compression of these models.

3.1.1. Weighted Single Stripification

The strip and connected-component merging operations explained above change the choice of the collapsible edges given by the minimum-weight perfect matching. Hence the final set of collapsible edges will be suboptimal. The goal is to limit this change in weight as much as possible while performing these merging operations.

We assign a cost to each possible nodal vertex processing, triangle-split, and edge-swap operations and sort them in a priority list [Pug90]. This cost is given by the increase in the overall weight of the matched edges after a certain operation. The cost of a nodal vertex processing would be the difference between the sums of the weights of the matched and unmatched edges around the vertex. The cost of a triangle-split operation is the cost of nodal vertex processing opera-

tion at the newly introduced vertex in the mid-point of the edge (refer Figure 2c). The cost of the edge swap operation is the difference between the cost of the original edge and the cost of the swapped edge with respect to the original model.

There are as many elements in the priority list as edges connecting separate loops. In general, many of these operations join the same pair of loops. When one operation is applied, it invalidates all its equivalent operations that are still in the list. Operations are popped from the front of the list and if they are valid, disjoint loops are merged. The algorithm terminates when there are no more operations to be processed or all the loops have been merged. This greedy algorithm also minimizes the total cost increase in choosing the new set of collapsible edges over the set that produces many collapsible-edge connected components.

3.2. Maintaining Strips During Simplification

In a view dependent simplification process, it helps to get additional performance boost by imposing a triangle strip structure over the simplified model at every frame. Maintaining classical triangle strips over a traditional simplification method that allows collapse of all edges has been studied earlier [ESAV99, SP03]. This requires special data structures and the strips tend to get shortened over the view-dependent simplification and refinement processes.

Hierarchyless simplification compensates for its restriction in the set of collapsible edges by facilitating trivial maintenance of strips. Since, by construction, the strip does not cross the collapsible edge, with every edge collapse, maintaining the strip involves removing the two triangles in either side of the collapsible edge from their respective strips and placing the previous and next triangles of the removed triangles next to each other in the strips; with every edge uncollapse, the corresponding triangles are inserted back. The total number of strips remains unchanged and traversing the new strip after every collapse/uncollapse operation is trivial. *Specifically, if the original model is represented using a single strip loop (for extreme simplification), every level of detail created by hierarchyless edge-collapse or uncollapse operation is represented using a single triangle strip loop.*

4. Collapsible Edges and Mesh Compression

Earlier we saw that the set of collapsible edges implicitly define triangle strip loops and enable us to maintain these strips through all the multi-resolution models during the hierarchyless simplification process. In this section, we focus on using these collapsible edges to efficiently encode the fundamental input information consisting of the mesh connectivity. Various techniques exist in the literature to compress the geometry, i.e. the vertex coordinates, which we will not address in this paper.

4.1. Traditional Region Growing Techniques

Due to the property of collapsible edges (for extreme simplification) that it implicitly defines a single triangle strip, any mesh compression technique such as [GS98, Ros99, RS99, Ise00] that is based on growing a connected mesh region from a start triangle and adding a single triangle at a time can be applied. Therefore, we could directly apply the Wrap&Zip compression technique [RS99] resulting in a guaranteed 1.7 bit per triangle encoding, or even smaller expected-case statistical coding. In fact, since we have a single triangle strip, the *split* code *S* in [RS99] can completely be avoided for genus-0 meshes leaving only 4 opcodes, and hence a guaranteed 2 bit per triangle encoding can be used. Handling complex meshes with handles follows the same argumentation as in [Ros99, RS99]. The use of a simple Edgebreaker [Ros99] or Wrap&Zip [RS99] code in conjunction with our single triangle-strip representation also compares well with [Ise00] that achieves similar connectivity encoding rates for encoding triangle strips.

Valence driven mesh encoding techniques [TG98, AD01] provide some improvement in compression rates over the above referenced methods. However, these do not allow for an arbitrary (triangle strip) mesh traversal and may thus lead to other extra costs.

4.2. Hand-and-Glove Mesh Compression

We now present a simple variant of a vertex-spanning tree based mesh encoding (see also [TR98]). The fundamental motivation for this algorithm is to encode the collapsible edge trees along with the connectivity between the trees, so that the compressed model can be viewed as the compression of the entire multi-resolution representation. This is also facilitated by the fact that in order to represent the multi-resolution models we do not need to represent or compress a hierarchy of collapses as in traditional simplification algorithm.

In genus zero manifold meshes we can see that a single triangle strip divides all vertices into two vertex-spanning trees, formed by the collapsible edges of our hierarchyless simplification. We call the two trees the *Hand* and the *Glove* as one conceptually wraps around the other in defining the triangle strip in between the two, see also Figure 4.

We first observe that both, the *Hand* and the *Glove* vertex-spanning trees can be represented by an arbitrary start node (indexed as 1 in Figure 4) and a depth- or breadth-first tree traversal. The traversal can be encoded by two bits per node: one bit indicating if there are any child nodes, and one bit indicating if the node has one or more siblings. With respect to Figure 4 this encodes the *Hand* starting at vertex 1 by the following sequence: 10, 11, 10, 11, 00, 00, 10, 10, 11, 00, 11, 00, 00. The same applies to the the *Glove* vertex-spanning tree. Since every vertex appears exactly once as a

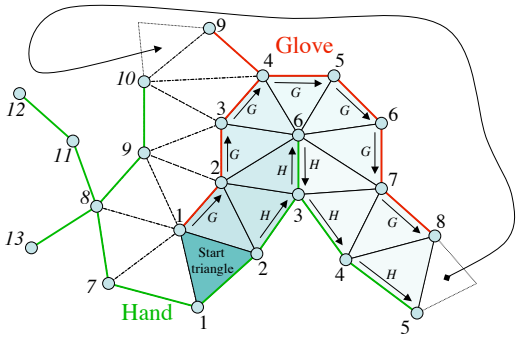

Figure 4: *Two connected components of collapsible edges forming 'Hand' and 'Glove' vertex spanning trees containing in-between them a single triangle strip loop traversing through the entire genus-0 manifold. Such a representation, implicitly given by extreme hierarchyless simplification, can be encoded in guaranteed 2 bits per triangle.*

node in either of the two trees we spend 2 bits per vertex, or equivalently 1 bit per triangle to encode this structure.

By definition of collapsible edges, every triangle has its collapsible edge part of either the *Hand* or the *Glove* tree and the third vertex in the opposite vertex-spanning tree. Therefore, the triangle strip is fully defined by a single start triangle and a sequence of instructions on how to advance to the next following triangle: taking the next vertex either from the *Glove* (*G*) or the *Hand* (*H*) vertex-spanning tree as shown in Figure 4. Advancing by *G* or *H* follows an out-order traversal along the corresponding vertex spanning tree. Hence for each triangle we need to specify *G* or *H* by one bit and thus the total representation cost amounts to a guaranteed 2 bit per triangle encoding.

Advantages of our approach not only include its simplicity and linear running time but also that there is no need to keep track of any extra information such as region boundaries or zipping rules as in prior approaches, and that the triangle strip formation is inherently part of our mesh representation and its encoding, and hence comes at no extra cost. Furthermore, the encoding efficiency is comparable to competitive approaches and leaves room for further optimization. Improved compression ratios can be achieved by combining the above basic coding scheme with pattern- or Entropy-based variable rate data compression techniques such as Lempel-Ziv [LZ78], Huffman [Huf52] or arithmetic coding [CNW87]. We report some initial results in Section 6 but further optimization of the encoding is beyond the scope of this paper and will be addressed by future extensions.

As presented, our encoding applies to genus zero manifold meshes because higher genus meshes may exhibit only a single vertex-spanning tree structure. However, this can easily be incorporated by virtually defining the *Hand* and *Glove* on the same vertex-spanning tree by different start points and initial traversal directions. Note that the vertex-spanning tree still only needs to be encoded once and thus no additional coding cost occurs besides some negligible initialization data.

5. Data Structure for Hierarchyless Simplification

In hierarchyless simplification, the connectivity around, and the geometric position of the new vertex formed after an edge collapse is determined the same way as in other edge-collapse based simplification algorithm. Further, the connectivity after edge uncollapses in any arbitrary order can be resolved as shown in Appendix A. The only issue that remains in hierarchyless simplification is that of finding the positions of two vertices after an edge uncollapse.

The positions of the two vertices resulting from an edge uncollapse is determined by the subsets of the collapsed vertices that they represent. Since, in conventional simplification methods the uncollapse order is determined by the collapse order, these positions are stored explicitly in the multiresolution hierarchy itself. In hierarchyless simplification, since the edges can be uncollapsed in any arbitrary order, it potentially requires exponential amount of memory to store the geometry for all possible permutations of orders of operations. We make use of the properties of hierarchyless simplification to present a new data structure that requires linear storage to solve this problem.

The set of collapsible edges form a forest as shown in Appendix A. The proposed data structure can be viewed as a variant of a half-edge data structure and handles each *collapsible tree* in the forest of collapsible edges independently. Consider a tree as shown in Figure 5. We construct a directed cycle called *collapsible cycle* around this tree by connecting the directed half-edges of the collapsible edges. We call the duplicates of the same vertex as *aliases* and the function *alias(i)* gives the next alias of i around the directed cycle.

We construct another tree, the *merged tree*, from the edges that are already collapsed. A directed *merged cycle* is constructed by connecting the duplicates and aliases in the *collapsible cycle* of the vertices belonging to the same *merged tree*. Note that every uncollapsed vertex in the *collapsible tree* is a merged tree and the *merged cycle* of this tree just connects the aliases of the vertex in order.

We store at each node i in the *collapsible cycle*, information about the segment of the merge cycle from $i+1$ till *alias(i)*. The data stored can be anything that helps to calculate the position of the vertex after an edge collapse. Further, the property of this stored data is that when cycles are split or merged, the data required for the resulting cycle(s) is computable from the existing data. For example, if the position of the resulting vertex is the average of all the collapsed vertex positions, then instead of storing this average, we store

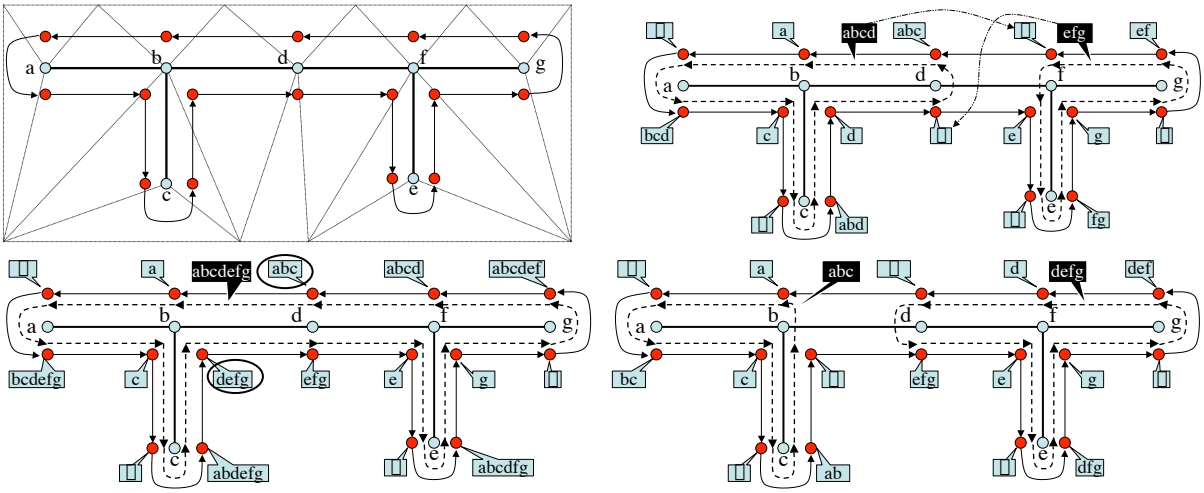

Figure 5: *Illustration of the Data Structure: (TOP-LEFT) Underlying triangulation and a tree of collapsible edges with vertices a-g. Every edge is represented as two directed half-edges connecting the representatives of the vertices of the edge. For example b has three representatives and are said to be* alias *of each other. Leaf nodes (aliases of a, c, e, g) are connected by end-cap directed edges. The cycle thus formed is called a* collapsible cycle. *(TOP-RIGHT) The vertices already collapsed (abcd and efg) form another set of cycles called the* merged cycles *(dashed directed edges). Every node in the collapsible cycle holds accumulated information about the vertices in the segment of the merged cycle till its next alias; if this segment has no vertex, then the information is empty (eg: one alias each of a, c, d, e, f,g). Notice that the union of the information contained in partner half-edges is actually the entire set. For example, the start node of the half edges of edge bc has information about c and abd, union of which is abcd - the entire merge cycle. Using this property, the amount of information stored can be exactly halved. (BOTTOM-LEFT)* ***Edge Collapse*** *of edge df: The entire merge cycle information of each of the merge cycles is copied to the nodes with empty information: efg to the empty alias of d and abcd to that of f. Individual merge cycle information are added together to get the final merge tree information from which the position of the vertex representing the entire tree can be calculated. Update of other nodes is done using a variant of union-find data structure lazily when required. (BOTTOM-RIGHT)* ***Edge Uncollapse*** *of edge bd: The information about the two merge cycles are in the start points of two half edges (circled in the bottom-left figure) of the collapsed edge using which the new vertex positions of individual merged trees are computed.*

the sum of the positions and the number of vertices independently. This enables weighted average calculation when two cycles are merged. In our implementation, we use accumulated quadrics [GH97] since we minimize the quadric error to find the position of the resulting collapsed vertex. The edge collapse and uncollapse operations using this data structure are described in Figure 5.

6. Implementation and Results

We have three parts to our implementation: (1) The first one is an off-line step to statically identify collapsible edges and find the minimum number of connected components of collapsible edges; in other words, a single triangle strip. (2) The second part is an online process of hierarchyless simplification that dynamically collapses and uncollapses any collapsible edges based on a given criterion. (3) The third step is again an off-line process to compress the topology of the collapsible edges and the stripification.

The implementation used to achieve (1) is based on the idea of graph matching, in our case a weighted-perfect matching solution as explained in Section 2, to find the set of collapsible edges. The current basic implementation makes use of the LEDA package to solve this task. To maximally join connected components of collapsible edge sets, nodal vertex processing, triangle split and edge swap operations are applied as outlined in Section 3.1. These steps are similar to [GE04] and as such define the single triangle strip. Finally, any cycles of collapsible edges are broken up by removing one of the collapsible edges in the cycle. The resulting vertex-spanning trees (at most two) based on collapsible edges are used for the hierarchyless simplification.

The input to (2) consists of the collapsible cycle(s) around these vertex-spanning tree(s), and an initial set of trivial merge cycles for each individual vertex. The dynamic maintenance of the collapsible and merge cycles as described in Section 5 is implemented using simple list data structures. To keep track of the current positions of collapsed vertex sets, which are defined by merge-cycles, is done via a union-find data structure. All edge collapse and uncollapse operations

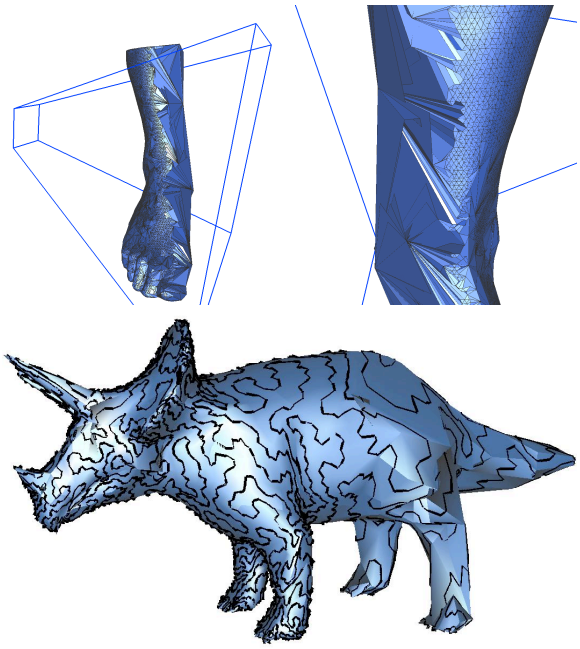

Figure 6: ***Top: (left)*** *View-dependent simplification of Foot model, originally with 53330 triangles.* ***(right)*** *Close-up of previous simplification.* ***Bottom:*** *Asymmetric simplification of trico model (rear is coarser than front), displaying the continuous triangle strip across different levels of detail.*

simply update the merge-cycles and the information attached to the nodes of the collapsible cycles.

As explained in Section 4.2, (3) incorporates a simple depth-first traversal of the vertex-spanning tree(s) and the triangle strip, sequentially outputting the corresponding codes for vertices and triangle faces. The start triangle and strip direction are implicitly given by chosing the *Hand* and *Glove* appropriately, as indicated in Figure 4, such that the first three vertices define it (one from the *Glove* and two from the *Hand*). In addition to the basic vertex and triangle bit-codes, we experimented with some simple variable-rate arithmetic coding as mentioned below.

Figure 7 shows different examples of progressive simplification sequences using our hierarchyless edge collapse operations. Despite the restricted set of edge-collapse operations, visual inspection of these experimental results shows comparable quality to that of models obtained with previous approaches. Additionally, in Figure 6 we show non-uniform and view-dependent simplification of back-facing triangles for a given view frustum, which demonstrates the power of unconstrained collapse/uncollapse operations that allow for dramatic changes in mesh resolution otherwise not easily achievable.

Below we report compression ratios achieved by our encoding method. As outlined in Section 4.2 we can guarantee a simple 2 bit per triangle encoding. Without any optimization, a simple adaptive arithmetic encoder [MNW98] on this raw bit-stream reduces the cost down to about 1.5 bits per triangle as shown in Table 1. This compares very well to other standard connectivity encodings such as an Edgebreaker implementation from [Shi] which we also followed by an arithmetic coding stage. Note that these experiments were performed on the meshes after joining connected components of collapsible edges and hence including the extra 3% triangles due to triangle splits.

Model	#Verts	bits/t HG+aac	bits/t EB	bits/t EB+aac	diff.
Trico	2894	1.73	1.98	1.62	06.9 %
Fandisk	6475	1.25	2.01	1.18	06.4 %
Blob	8036	1.61	2.01	1.46	10.5 %
Balljoint	137062	1.5	2.0	1.28	15.2 %
Armadillo	177354	1.56	N/A	N/A	N/A

Table 1: *Mesh compression results using our Hand-and-Glove encoding (HG) compared to Edgebreaker (EB), both including an adaptive arithmetic coder (aac) stage. Cost is reported as bits-per-triangle for mesh connectivity.*

7. Summary

We have presented a completely different and a novel perspective of geometric simplification through the prism of stripification and geometric compression. In this process, we have identified and explored an interesting and exciting algorithmic space between all these three apparently different fields and problems. The relationship between these fields has been lucidly brought out by just a subset of edges, what we call the *collapsible edges*, of the model. The existence of collapsible edges is not just theoretical, but by relating this to more than a century old graph matching problem and its solution, we have made a practical contribution to the field of geometric simplification. We saw interesting topological results that even with these restricted set of collapsible edges, we can perform extreme simplification of the model. Further, any short-coming in the quality of simplification is compensated by the ability to represent any simplified model using a single triangle strip. Further, the compression of the multi-resolution models of hierarchical simplification has the same complexity as of compressing the original model. Since the technique lies in the intersection space of simplification, stripification, and compression, it can be used to develop systems that has both efficient off-line storage and efficient online-rendering of models.

As part of the future work, we would like to extend to models with boundaries and apply our compression technique to models with higher genus boundaries. The same common space in higher dimensions can also be explored.

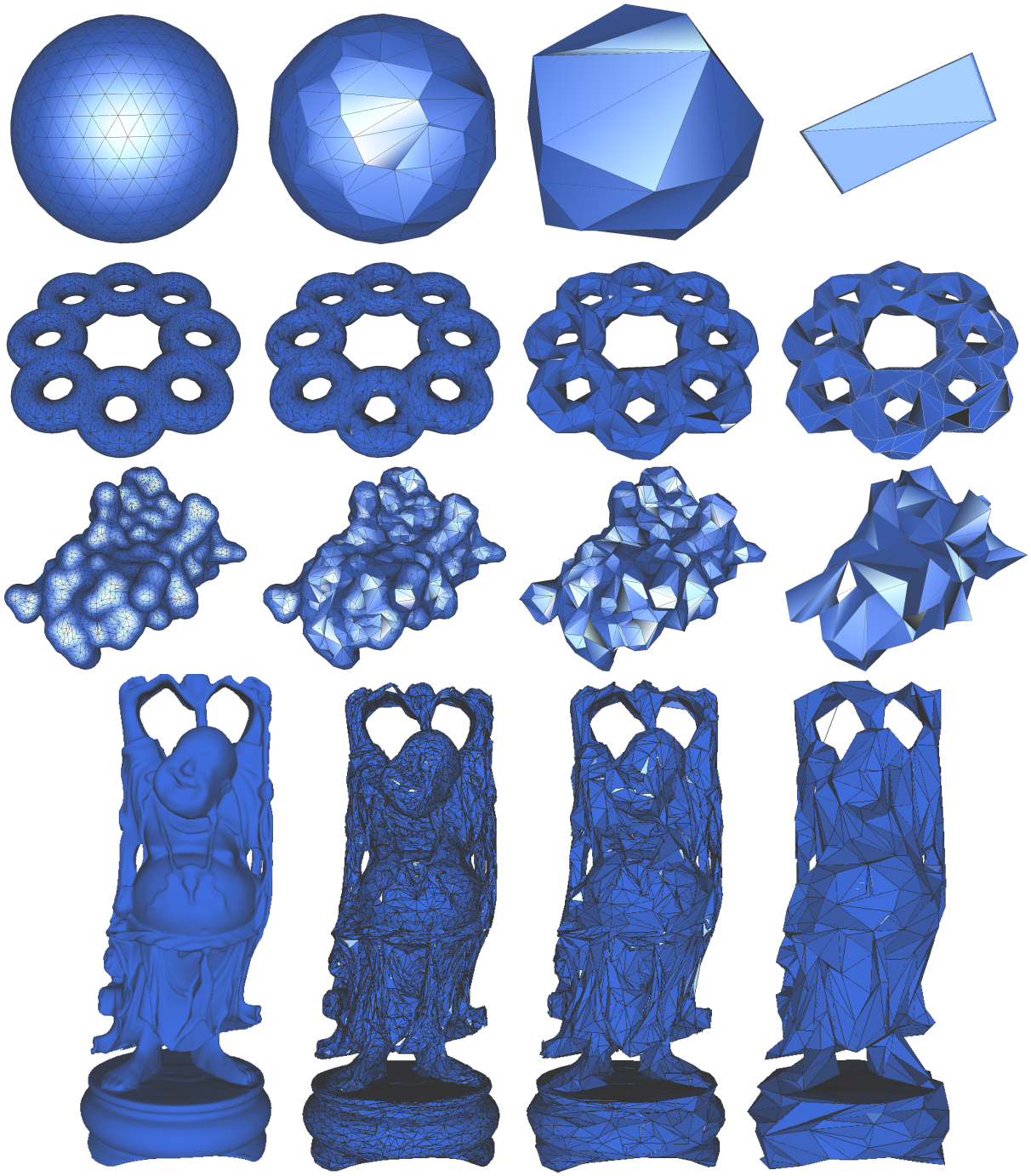

Figure 7: *Progressive simplification of 4 models. From top to bottom:* ***Sphere:*** *Models with 1358, 454, 54 and 4 triangles. Note the* extremal simplification *to a tetrahedron.* ***9-torus:*** *Models with 19778, 7238, 1500 and 778 triangles.* ***Blob:*** *Models with 16450, 6450, 2450 and 450 triangles.* ***Happy:*** *Models with 101924, 33924, 9924 abd 1924 triangles.*

References

[AD01] ALLIEZ P., DESBRUN M.: Valence-driven connectivity encoding for 3D meshes. *Computer Graphics Forum (EUROGRAPHICS) 20*, 3 (2001), 480–489. 5

[BBDL01] BIEDL T. C., BOSE P., DEMAINE E. D., LUBIW A.: Efficient algorithms for Petersen's matching theorem. *J. Algorithms 38* (2001), 110–134. 3

[BRR*01] BELMONTE O., REMOLAR I., RIBELLES J., CHOVER M., REBOLLO C., FERNANDEZ M.: Multiresolution triangle strips. In *Proceedings IASTED Invernational Conference on Visualization, Imaging and Image Processing* (2001), pp. 182–187. 1

[Cho97] CHOW M. M.: Optimized geometry compression for real-time rendering. In *Proc. IEEE Visualization* (1997), pp. 347–354. 1

[CNW87] CLEARY J. G., NEAL R. M., WITTEN I. H.: Arithmetic coding for data compression. *Communications of the ACM 30*, 6 (June 1987), 520–540. 6

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-perserving simplification. In *SIGGRAPH* (1998), ACM Press, pp. 115–122. 1

[Dee95] DEERING M.: Geometry compression. In *Proceedings SIGGRAPH* (1995), pp. 13–20. 1

[DP02] DECORO C., PAJAROLA R.: XFastMesh: Fast view-dependent meshing from external memory. In *Proceedings IEEE Visualization* (2002), pp. 363–370. 1

[ESAV99] EL-SANA J., AZANLI E., VARSHNEY A.: Skip strips: Maintaining triangle strips for view-dependent rendering. In *Proc. IEEE Visualization* (1999), pp. 131–138. 1, 5

[ESC00] EL-SANA J., CHIANG Y.-J.: External memory view-dependent simplification. In *Proceedings EUROGRAPHICS* (2000), pp. 139–150. 1

[ESV96] EVANS F., SKIENA S., VARSHNEY A.: Optimizing triangle strips for fast rendering. In *Proceedings IEEE Visualization* (1996), pp. 319–326. 1

[ESV99] EL-SANA J., VARSHNEY A.: Generalized view-dependent simplification. In *Proc. EUROGRAPHICS* (1999), pp. 83–94. 1

[GE04] GOPI M., EPPSTEIN D.: Single strip triangulation of manifolds with arbitrary topology. *Computer Graphics Forum (EUROGRAPHICS) 23*, 3 (2004), 371–379. 4, 7

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings SIGGRAPH* (1997), pp. 209–216. 1, 7

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization* (1998), pp. 264–269. 1

[Gop04] GOPI M.: Controllable single-strip generation for triangulated surfaces. In *Pacific Graphics* (2004), IEEE Computer Society, pp. 61–69. 1

[GS98] GUMHOLD S., STRASSER W.: Real time compression of triangle mesh connectivity. In *Proceedings SIGGRAPH* (1998), pp. 133–140. 5

[HDD*93] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Mesh optimization. In *Proceedings SIGGRAPH* (1993), pp. 19–26. 1

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings SIGGRAPH* (1996), pp. 99–108. 1

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *SIGGRAPH* (1997), pp. 189–198. 1

[Huf52] HUFFMAN D. A.: A method for the construction of minimum redundancy codes. In *Proceedings Inst. Electr. Radio Eng.* (1952), pp. 1098–1101. 6

[Ise00] ISENBURG M.: Triangle strip compression. In *Graphics Interface 2000* (2000), pp. 197–204. 1, 5

[KL03] KIM J., LEE S.: Transitive mesh space of a progressive mesh. *IEEE Trans. on Visualization and Computer Graphics 9*, 4 (Oct-Dec 2003), 463–480. 1

[Kor99] KORNMANN D.: Fast and simple triangle strip generation. Technical Report, 1999. 1

[LT98] LINDSTROM P., TURK G.: Fast and memory efficient polygonal simplification. In *IEEE Visualization* (1998), pp. 279–286. 1

[Lue01] LUEBKE D. P.: A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics & Applications 21*, 3 (May/June 2001), 24–35. 1

[LZ78] LEMPEL A., ZIV J.: Compression of individual sequences via variable-rate coding. *IEEE Trans. on Information Theory 24*, 5 (September 1978), 530–536. 6

[MNW98] MOFFAT A., NEAL R., WITTEN I. H.: Arithmetic coding revisited. *ACM Transactions on Information Systems 16*, 3 (July 1998), 256–294. 8

[Paj01] PAJAROLA R.: Fastmesh: Efficient view-dependent meshing. In *Proceedings Pacific Graphics* (2001), pp. 22–30. 1

[Pet91] PETERSON J. P. C.: Die theorie der regularen graphs (The Theory of Regular Graphs). *Acta Mathematica 15* (1891), 193–220. 3

[Pug90] PUGH W.: Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM 33*, 6 (1990), 668–676. 4

[Ros99] ROSSIGNAC J.: Edgebreaker: Compressing the incidence graph of triangle meshes. *IEEE Trans. on Visualization and Computer Graphics 5*, 1 (January-March 1999), 47–61. 5

[RS99] ROSSIGNAC J., SZYMCZAK A.: Wrap&zip decompression of the connectivity of triangle meshes compressed with edgebreaker. *Journal of Computational Geometry, Theory and Applications 14*, 1-3 (November 1999), 119–135. 5

[Shi] SHIKHARE D.: Edgebreaker 3D compression package. http://www.angelfire.com/space2/dineshshikhare/. 8

[SP03] SHAFAE M., PAJAROLA R.: DStrips: Dynamic triangle strips for real-time mesh simplification and rendering. In *Pacific Graphics* (2003), pp. 271–280. 1, 5

[Ste01] STEWART A. J.: Tunneling for triangle strips in continuous level-of-detail meshes. In *Proceedings Graphics Interface* (2001), pp. 91–100. 1

[TDG*00] TAUBIN G., DEERING M., GOTSMAN C., GUMHOLD S., ROSSIGNAC J.: 3D geometry compression. SIGGRAPH 2000 Course Notes 38, 2000. 1

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Proceedings Graphics Interface* (1998), pp. 26–34. 5

[Tho00] THORUP M.: Near-optimal fully-dynamic graph connectivity. In *Proc. 32nd Annu. ACM Symp. on Theory of Computing* (2000), pp. 343–350. 3

[TR98] TAUBIN G., ROSSIGNAC J.: Geometric compression through topological surgery. *ACM Transactions on Graphics 17*, 2 (1998), 84–115. 5

[XHM99] XIANG X., HELD M., MITCHELL J. S. B.: Fast and effective stripification of polygonal surface models. In *Proceedings Sym. Interactive 3D Graphics* (1999), pp. 71–78. 1

[XV96] XIA J. C., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. In *Proceedings IEEE Visualization* (1996), pp. 327–334. 1

Appendix A: Proof of Uniqueness of Adjacency in Hierarchyless Simplification Operations

Observation: Consider a *connected graph G* of collapsible edges and the vertices they connect. Construct a spanning tree *S* of this graph. Since both *G* and *S* are connected graphs, collapsing all the edges will result in a single vertex in both the graphs. Following the same reason, a spanning tree of every connected component of collapsible edges can be computed such that collapsing the edges in these spanning trees will result in the same number of vertices as collapsing the original set of edges. Hence, we can assume that the set of collapsible edges is a forest of connected components.

Theorem 3 In a triangulated two manifold whose set of collapsible edges is a forest of connected components, the connectivity change due to an arbitrary edge uncollapse can be uniquely determined.

Proof If the set of collapsible edges form a forest of connected components, so does the set of already merged edges which is actually a subset of the set of collapsible edges. The uncollapse operations can be performed only on the edge that is already merged (collapsed). Identify the (merged) tree to which the vertices of the uncollapsed edge belong. A tree is one connected; removing one edge will disconnect the tree. Hence, given the edge *AB*, we can partition the vertices of the tree into two groups, say left and right, one on the side of *A* and the other on the side of *B*, and the new vertex positions of these two groups can be computed unambiguously. Once the positions of the (representatives of the) vertices are found, resolving the connectivity is trivial: if there is an edge between two vertices *a* and *b* in the original mesh, connect the representatives of *a* and *b* in the simplified mesh with an edge. Similarly, face connectivity can also be resolved. □

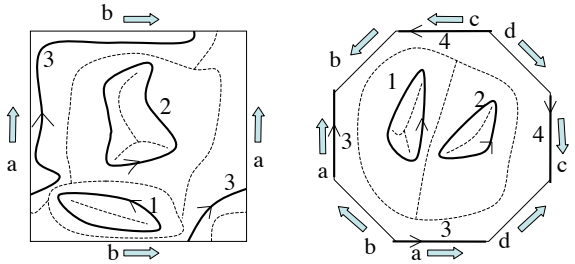

Figure 8: *Planar unfoldings of a torus and a double-torus. Every alternate edge can be a non-contractible non-intersecting closed loop. The number of connected regions on the manifold depends only on the number of contractible closed curves. In both figures, curves 1 and 2 are contractible and others are not. Both these manifolds have three connected regions.*

Appendix B: Proof of Theorem 2

Theorem: If s triangle strip loops cover the entire triangulated two manifold of genus g then the number of connected collapsible edge components $c > 0$ and $s+1-g \leq c \leq s+1$.

Proof The triangle strip loops on a manifold neither self-intersect nor intersect with each other. Hence they can be considered as closed curves on the surface. The number of connected components of the 'medial axis' collapsible edges is same as the number of regions the manifold is partitioned by the closed triangle strip curves. On a plane, the number of connected components is $n+1$ for n closed curves. This is true for a sphere also. For a higher genus object, while every contractible closed curve partitions the space into two, non-contractible closed curves do not. Figure 8 shows planar unfolding of a torus and a double torus. In both these unfoldings, curves 1 and 2 are contractible closed curves and other curves are non-contractible curves homotopic to fundamental curves. Even though there are $2g$ fundamental closed curves, where g is the genus, only g of them are mutually non-intersecting. Hence there is a minimum of $n-g$ contractible closed curves, and the number of connected partitions is $n+1-g$. □