Computing a Canonical Polygonal Schema of an Orientable Triangulated Surface

Lazarus, Francis; Pocchiola, Michel; Vegter, Gert; Verroust, Anne

Published in:
EPRINTS-BOOK-TITLE

Publication date:
2001

Computing a Canonical Polygonal Schema of an Orientable Triangulated Surface

Francis Lazarus[*]
Michel Pocchiola[†]
Gert Vegter[‡]
Anne Verroust[𝕊]

ABSTRACT

A closed orientable surface of genus g can be obtained by appropriate identification of pairs of edges of a $4g$-gon (the polygonal schema). The identified edges form $2g$ loops on the surface, that are disjoint except for their common end-point. These loops are generators of both the fundamental group and the homology group of the surface. The inverse problem is concerned with finding a set of $2g$ loops on a triangulated surface, such that cutting the surface along these loops yields a (canonical) polygonal schema. We present two optimal algorithms for this inverse problem. Both algorithms have been implemented using the CGAL polyhedron data structure.

1. INTRODUCTION

Let M_g be a regular $4g$–gon, whose successive edges are labeled $a_1, b_1, \overline{a}_1, \overline{b}_1, \cdots, a_g, b_g, \overline{a}_g, \overline{b}_g$. Edge x is directed counterclockwise, edge $\overline{x}$ clockwise. The space obtained by identifying edges x and $\overline{x}$, as indicated by their direction, is a closed oriented surface; See e.g. [8, Chapter 1.4]. This surface, called orientable surface of genus g, is homeomorphic to a 2–sphere with g handles. E.g., $\mathcal{M}_1$ is the torus; See Figure 1. The labeled polygon M_g is called the *canonical polygonal schema* of $\mathcal{M}_g$.

It is easy to see that all vertices are identified to a single point p_0 of the surface. After identification in pairs, the edges of the polygonal schema form $2g$ curves on $\mathcal{M}_g$, which are disjoint, except for their common endpoint p_0. These $2g$ loops are generators of the fundamental group of $\mathcal{M}_g$ (and of the first homology group). In the sequel we drop the dependence on the genus from our notation, i.e., $\mathcal{M}$ denotes a closed orientable surface of genus g.

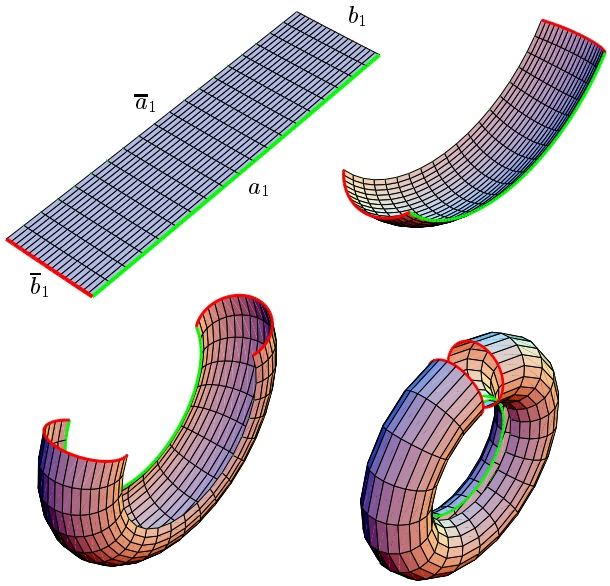

Figure 1: From polygonal schema to orientable surface: the torus.

In this paper we consider the inverse problem: Given a combinatorial (triangulated) surface, find a *canonical* set of PL-curves (generators) such that, after cutting the surface along these generators, we obtain a canonical polygonal schema for the surface. A PL-curve is an alternating sequence of edges and vertices, where edges connect two successive vertices that lie in the same face, either in its interior or on the interior of one of its boundary edges.

In [10] an algorithm is sketched that constructs a canonical set of generators in optimal time and space. In this paper, we present in detail a simple optimal algorithm; we call this the *incremental method*, since we construct the generators while traversing all triangles of the surface. Our main result is

[*]CNRS and University of Poitiers, France.
E-mail: lazarus@sic.sp2mi.univ-poitiers.fr
[†]Dépt d'Informatique, Ecole Normale Supérieure, Paris, France. E-mail: Michel.Pocchiola@ens.fr
[‡]Dept. of Math. and CS, University of Groningen, The Netherlands. E-mail: gert@cs.rug.nl
[𝕊]INRIA Rocquencourt, France.
E-mail: Anne.Verroust@inria.fr

SCG'01, June 3-5, 2001, Medford, Massachusetts, USA.

THEOREM 1. *A canonical set of PL-generators for an orientable closed surface of genus g, with a total of n vertices, edges and faces, can be computed in $O(gn)$ time and space, which is worst-case optimal. Each PL-generator consists of $O(n)$ edges and vertices.*

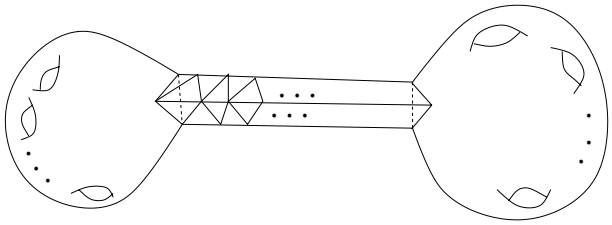

Figure 2: A surface with two groups of $\lceil g/2 \rceil$ and $\lfloor g/2 \rfloor$ handles, separated by a thin tunnel of size $\Omega(n)$. Regardless of the position of the base-point p_0, at least half of this tunnel must be crossed by at least $\lfloor g/2 \rfloor$ generators.

Optimality is easy to establish; See Figure 2.

Furthermore, we show how to turn Brahana's method [2] into a second algorithm computing a canonical set of generators in optimal time and space. We have implemented both methods using the `C++` library CGAL. For comments on these implementations, and their performance, we refer to Section 6.

There are several reasons for presenting these algorithms here: (i) our algorithms greatly simplify the method of [10], (ii) full details are presented for the first time, (iii) the algorithms have been implemented, and (iv) the algorithms can be used to solve several other problems in computational topology. Among the applications are the construction of PL-homeomorphisms between surfaces, and the construction of (a part of) the universal covering space of the surface. A similar, non-canonical polygonal schema has been used in [6] to decide whether two PL-curves on a surface are homotopic. A different algorithm for the latter problem, based on methods from combinatorial group theory, and abandoning universal covering spaces, is presented in [4]. Other applications are conceivable in connection with morphing, where a suitable parametrization of 2-manifolds is provided by the disk obtained by cutting along the canonical generators.

For general background material on computational topology, also in connection with applications, we refer to the surveys [5] and [9].

2. SURFACES WITH COLLARS

Triangulated surfaces will be represented by Doubly-Connected Edge List, a data structure for representing subdivisions of surfaces. We refer to [3, Chapter 2] for details on this data structure. Note that every undirected edge of the triangulation corresponds to exactly two half-edges. The incremental algorithm starts with the open surface $\mathcal{S} = \mathcal{M} \setminus \{t_0\}$, where t_0 is an arbitrary (closed) triangle, eventually containing the common base point of the constructed generators. Initially, the topological boundary $\mathcal{B}$ of $\mathcal{S}$ is the boundary of t_0. The algorithm proceeds by visiting triangles incident to $\mathcal{B}$ along at least one edge, and cutting these (closed) triangles from $\mathcal{S}$. Note that the non-visited part of $\mathcal{M}$ is an open subset of $\mathcal{M}$. The topological boundary $\mathcal{B}$ is adjusted accordingly. It is represented as a circular sequence of half-edges, oriented in such a way that the triangle to the left of a half-edge belongs to $\mathcal{S}$. We say that a vertex *occurs* in $\mathcal{B}$ if it is the origin of a half-edge in $\mathcal{B}$.

As we will explain in more detail, the boundary $\mathcal{B}$ may become non-regular during this process, in the sense that a vertex occurs multiply in $\mathcal{B}$, or it contains both a half-edge and its opposite partner (called its *Twin* in [3]). See Figure 3 (Bottom). Yet, the irregularity of $\mathcal{B}$, and hence of the surface $\mathcal{S}$, is restricted. This is made more precise by introducing the notion of a *collar* of an open surface.

DEFINITION 2. *A surface with collar in $\mathcal{M}$ is a pair $(\mathcal{S}, c)$, where $\mathcal{S}$ is an open submanifold of $\mathcal{M}$, and $c : \mathbb{S}^1 \times [0,1] \to \mathcal{M}$ is a continuous map, such that*
1. $c(\mathbb{S}^1 \times (0,1]) \subset \mathcal{S}$, and the restriction $c\,|_{\mathbb{S}^1 \times (0,1]} : \mathbb{S}^1 \times (0,1] \to \mathcal{S}$ is an embedding;
2. $c(\mathbb{S}^1 \times \{0\}) \subset \mathcal{M} \setminus \mathcal{S}$;
3. The topological boundary of $\mathcal{S}$ (viz $\overline{\mathcal{S}} \setminus \mathcal{S}$) is the image of the closed curve $c : \mathbb{S}^1 \times \{0\} \to \mathcal{M}$.

Observe that the curve $c : \mathbb{S}^1 \times \{0\} \to \mathcal{M}$ is in general *not* an embedding. The curve $c : \mathbb{S}^1 \times \{1\} \to \mathcal{M}$, which is an embedding, may be considered as a 'regularization' of the – perhaps non-regular – boundary of $\mathcal{S}$. We refer to the half-open strip $c(\mathbb{S}^1 \times (0,1])$ as the *collar of* $\mathcal{S}$. This collar has *attachment curve* $c(\mathbb{S}^1 \times \{0\})$, and *free boundary* $c(\mathbb{S}^1 \times \{1\})$. Note that every continuous curve connecting a point in $\mathcal{S}$ with a point in $\mathcal{M} \setminus \mathcal{S}$ intersects the collar of $\mathcal{S}$.

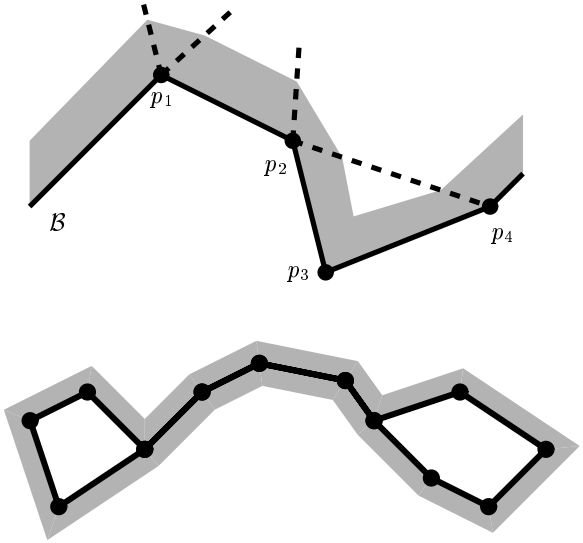

Figure 3: Collars. Top: a PL-collar is obtained by inserting vertices near the tail of half-edges incident to $\mathcal{B}$, or in a corner of a triangle. Bottom: a collar on a singular curve $\mathcal{B}$.

A collar $\mathcal{S}$ has a straightforward representation in the

PL-setting. To this end, we insert a vertex near the tail of each half-edge in $\mathcal{S}$ emanating from a vertex of $\mathcal{B}$. Note that in this way an edge with both endpoints on $\mathcal{B}$ obtains two vertices. Furthermore, if two successive half-edges of $\mathcal{B}$, sharing a common vertex v, are incident to the same triangle t of $\mathcal{S}$, there is no half-edge of $\mathcal{S}$ emanating from v. In this case, we insert a vertex in the interior of t (e.g., on the bisector of the angle of t at v). Connecting the sequence of inserted vertices by edges we obtain a PL-collar of $\mathcal{S}$; See Figure 3. This type of collar will be used in Section 4.

As usual, the Euler characteristic $\chi(\mathcal{S})$ of $\mathcal{S}$ is the alternating sum of the numbers of vertices, edges and faces of $\mathcal{S}$. Cutting the surface along $\mathcal{B}$ we obtain a boundary of $\mathcal{S}$ consisting of a cyclic sequence of half-edges (where some pairs of half-edges may correspond to the same undirected edge of $\mathcal{M}$). Gluing a disk along this cyclic sequence of half-edges yields a closed orientable surface. By definition, the genus g of $\mathcal{S}$ is the genus of the latter surface. It is straightforward to check that $\chi(\mathcal{S}) = 1 - 2g$.

3. OUTLINE OF THE ALGORITHM

We now describe the algorithm that visits all triangles of $\mathcal{M}$, starting from a single triangle. This algorithm is the backbone for the construction of a canonical system of generators, to be described in Section 4. Globally speaking the algorithm proceeds as follows. The algorithm ConnectedSum, which is called on the complement $\mathcal{S}$ of the initial triangle, visits a triangle t incident upon the topological boundary $\mathcal{B}$ of $\mathcal{S}$, updates $\mathcal{S}$ and $\mathcal{B}$, and calls itself recursively on the updated version of $\mathcal{S}$. (As we shall explain at the end of this section, the algorithm in fact decomposes the surface $\mathcal{M}$ as a connected sum of tori, whence its name.) During this recursive process, $\mathcal{S}$ may become disconnected, in which case ConnectedSum is called recursively on each connected component. It may also happen that $\mathcal{S}$ is not disconnected, but is not a surface with collar either (it will turn out that in the latter case the collar is split). Before we present the algorithm in full detail in Figure 5, we first specify the input of the algorithm.

Precondition of ConnectedSum**.** Algorithm ConnectedSum takes as input a pair $(\mathcal{S}, g)$, where $\mathcal{S}$ is a surface with collar, which has *positive* genus g.

In particular, the condition $g > 0$ guarantees that ConnectedSum will not be called on disks, which will be crucial in the analysis of the time complexity. The process of visiting triangle t, incident upon the topological boundary $\mathcal{B}$, is called an *extension*. We distinguish two types of extensions.

Regular Extension: Triangle t shares either two vertices and one half-edge h_1 (Figure 4, top), or three vertices and two half-edges h_1, h_2 (Figure 4, bottom), with $\mathcal{B}$.

We update $\mathcal{B}$ in the former case by replacing the half-edge h_1 with the two-chain h_2, h_3, in the latter case by replacing the two-chain h_1, h_2 with the half-edge h_3. Note that the topological types of $\mathcal{B}$ and the collar do not change upon a regular extension. In particular,

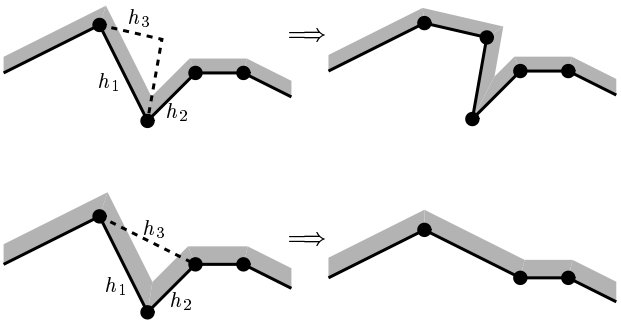

Figure 4: A regular extension.

$\mathcal{S}' = \mathcal{S} \setminus t$ is a surface with collar (to guarantee that $\mathcal{S}'$ is an open subset of $\mathcal{M}$, we consider the triangle t to be closed). Therefore, ConnectedSum is called recursively on $\mathcal{S}'$. It is obvious that the Euler characteristic, and hence the genus, does not change under regular extension.

Splitting Extension: Triangle t shares three vertices and one half-edge with $\mathcal{B}$ (Figure 6, upper part).

The vertex of t, not adjacent to the common half-edge of $\mathcal{B}$ and t, is called the *split vertex*, and is denoted by v_s. Let the vertices of t be v_1, v_2 and v_3, such that v_1v_2 is a half-edge of $\mathcal{B}$, and hence $v_3 = v_s$. Let L be the part of $\mathcal{B}$ between v_3 and v_1, and let R be the part between v_2 and v_3. Then $\mathcal{B}$ is split into $\mathcal{B}_l = v_1v_3L$ and $\mathcal{B}_r = v_3v_2R$. We distinguish two sub-cases:

$\mathcal{S} \setminus t$ is not connected. In this case $\mathcal{S} \setminus t$ consists of two connected components, $\mathcal{S}_l$ and $\mathcal{S}_r$ say, with topological boundary $\mathcal{B}_l$ and $\mathcal{B}_r$, respectively. Both $\mathcal{S}_l$ and $\mathcal{S}_r$ are surfaces with collars, with attachment curves $\mathcal{B}_l$ and $\mathcal{B}_r$, respectively.

$\mathcal{S} \setminus t$ is connected. In this case the topological boundary of $\mathcal{S} \setminus t$ is $\mathcal{B}_l \cup \mathcal{B}_r$, so $\mathcal{S} \setminus t$ is not a surface with collar. In particular, ConnectedSum does not accept $\mathcal{S} \setminus t$ as input. To remedy this situation, let γ be a simple edge-path in $\mathcal{S} \setminus t$ connecting $\mathcal{B}_l$ and $\mathcal{B}_r$, called a *join-path* (of $\mathcal{B}_l$ and $\mathcal{B}_r$). See Figure 6, where $v_l \in \mathcal{B}_l$ and $v_r \in \mathcal{B}_r$ are the extremal vertices of γ.

The following result, whose (straightforward) proof is omitted from this version of the paper, guarantees that in case of a splitting extension the ConnectedSum can be called recursively:

Lemma 3. *Suppose processing t causes a splitting extension.*
1. If $\mathcal{S}\setminus t$ is connected, and γ is a join-path, then $\mathcal{S}\setminus(t\cup\gamma)$ is a surface with collar, having genus $g-1$.
2. If $\mathcal{S} \setminus t$ is not connected, its connected components $\mathcal{S}_l$ and $\mathcal{S}_r$ are surfaces with collar. Moreover, if their genuses are g_l and g_r, respectively, then $g = g_l + g_r$.

The algorithm that constructs a canonical set of generators is presented in Figure 5. Checking whether the current extension is regular (line 2) can be done in $O(1)$ time, by setting a mark bit for each visited vertex. To determine whether $\mathcal{S}' = \mathcal{S} \setminus t$ is connected, we try to construct a join-path γ by performing a breadth-first search on the 1-skeleton of $\mathcal{S}'$ (we have to say more

ConnectedSum($\mathcal{S}$)
1 Let t be a triangle of S incident to collar($\mathcal{S}$)
2 **if** t causes regular extension
3 **then** ConnectedSum($\mathcal{S} \setminus t$)
4 **else** $\triangleleft$ collar($\mathcal{S} \setminus t$) is disconnected
5 **if** $\mathcal{S} \setminus t$ is connected
6 **then** construct PL-path γ joining components of collar($\mathcal{S} \setminus t$)
7 construct pair of generators along γ and part of collar($\mathcal{S} \setminus t$)
8 **if** genus($\mathcal{S}$) > 1
9 **then** ConnectedSum($\mathcal{S} \setminus (t \cup \gamma)$)
10 **else** let $\mathcal{S}_l$ and $\mathcal{S}_r$ be the components of $\mathcal{S} \setminus t$
11 **if** genus($\mathcal{S}_l$) > 0 **then** ConnectedSum($\mathcal{S}_l$)
12 **if** genus($\mathcal{S}_r$) > 0 **then** ConnectedSum($\mathcal{S}_r$)

Figure 5: Algorithm ConnectedSum

about this presently). Depending on whether we succeed in connecting the two components $\mathcal{B}_l$ and $\mathcal{B}_r$ of the topological boundary of $\mathcal{S}'$, we decide whether $\mathcal{S}'$ is connected or not. If $\mathcal{S}'$ is connected, a pair of generators is constructed (details are presented in Section 4), and ConnectedSum is recursively called on $\mathcal{S} \setminus (t \cup \gamma)$, if $g' = g - 1 > 0$. If $\mathcal{S}'$ is not connected, ConnectedSum is recursively called on the connected components $\mathcal{S}_l$, if $g_l > 0$, and $\mathcal{S}_r$, if $g_r > 0$.

Lemma 4. *1. If $\mathcal{S} \setminus t$ is connected, establishing connectedness and computing a join-path γ can be performed in time proportional to the size of $\mathcal{S}$.*
2. If $\mathcal{S} \setminus t$ has two connected components, establishing non-connectedness and computing the genuses of the connected components can be performed in time proportional to the size of the smaller connected component.

We only give a sketch of the proof. When a split occurs, we try to construct the join-path γ by means of a *tandem search* traversing the edges of the surface in parallel, starting from the sources $\mathcal{B}_l$ and $\mathcal{B}_r$. Then either the tandem search succeeds in connecting $\mathcal{B}_l$ and $\mathcal{B}_r$ by the join-path γ, or it detects that $\mathcal{S} \setminus t$ has two connected components $\mathcal{S}_l$ and $\mathcal{S}_r$ by exhausting the smaller of these two components. In the latter case we compute the genus of the smaller component by determining the number of vertices, edges and faces. Lemma 4, part 2, gives the genus of the other connected component.
Lemma's 3 and 4 allow us to analyze the time complexity of the traversal of the initial surface $\mathcal{M}$. To this end, let t_0 be an arbitrary triangle of $\mathcal{M}$.

Corollary 5. *The call of* ConnectedSum *on the surface with collar $\mathcal{S}_0 = \mathcal{M} \setminus t_0$ is executed in time $O(gn)$, plus the time needed to construct the g pairs of (line 7) generators upon a non-disconnecting split. Here g is the genus of $\mathcal{M}$ and n is the total number of vertices, edges and triangles in $\mathcal{M}$.*

Again the proof is straightforward, except for a minor subtlety. If all recursive calls either result in the construction of a pair of generators (line 5–9), or in recursive calls of ConnectedSum on collared surfaces of lower genus (i.e., $g_l > 0$ and $g_r > 0$), the total time complexity obviously is $O(gn)$ (since the total number of calls is g in this lucky case).

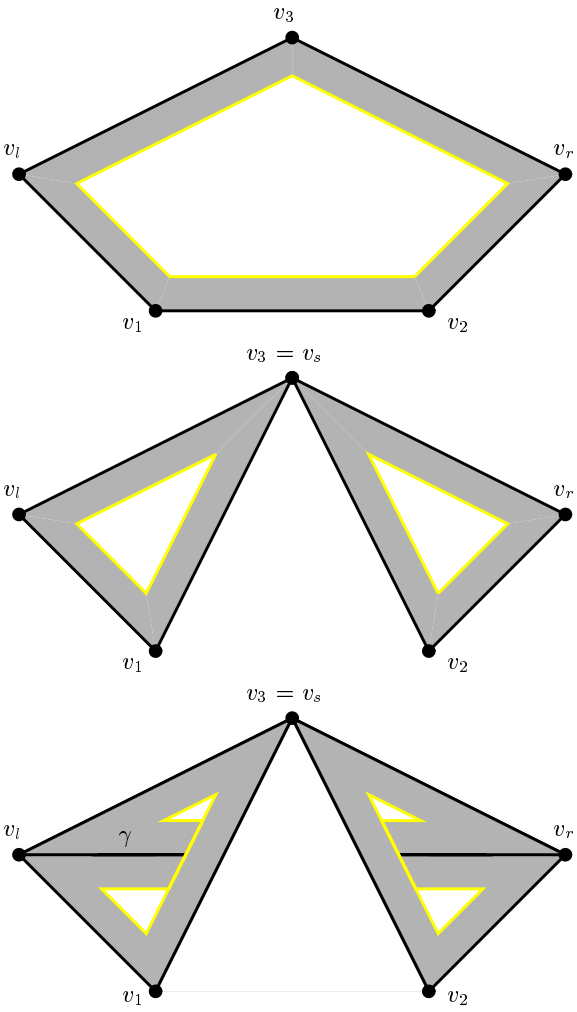

Figure 6: A splitting extension.

However, ConnectedSum may be called recursively on a collared surface of the same genus as $\mathcal{S}$ in case the genus of $\mathcal{S}_l$ or $\mathcal{S}_r$ is zero. Note that in this case the component with genus zero is discarded. Since the algorithm spends time proportional to the smaller of the sizes of $\mathcal{S}_l$ or $\mathcal{S}_r$, we charge the cost of the recursive call to the discarded component. Therefore, the total cost of calls of this type is $O(n)$.

4. CONSTRUCTING GENERATORS

It remains to fill in the details of the construction of a pair of generators, cf Figure 5, line 7. These generators will be routed along an *approach path* γ_{AP}, which connects the base point with the boundary of the non-visited part of the surface. As the algorithm proceeds,

we should take care that generators we are about to complete do not intersect already constructed generators. Yet, we allow already constructed generators to intersect the non-visited part of the surface, although possible intersections should be confined to the collar of the non-visited part.

More precisely, let t_0 be the first triangle visited, and let the base-point p_0 be an interior point of t_0. We first extend the precondition, introduced in Section 2 for calling ConnectedSum on a non-visited surface $\mathcal{S}$ with collar. To this end, we assume from now on that a collar is piecewise linear, as described in Section 2 (See also Figure 3). In particular, a collar of $\mathcal{S}$ only intersects edges and faces of $\mathcal{S}$ incident upon the attachment curve $\mathcal{B}$, and such edges are intersected in interior points. Furthermore, we require that the attachment curve $\mathcal{B}$ of $\mathcal{S}$ has a distinguished half-edge h_{APA}, satisfying the following conditions:

(AP1) The base-point p_0 is connected by a PL-curve γ_{AP} to h_{APA}; apart from p_0, this *approach path* is disjoint from $\mathcal{S}$, and it does not share any point with already constructed generators and approach paths;

(AP2) The terminal point of γ_{AP} on h_{APA} can be connected to the free boundary of the collar of $\mathcal{S}$ by a line segment inside the face of $\mathcal{S}$ incident upon h_{APA}, which does not intersect any of the generators constructed so far;

(AP3) No already constructed generator intersects the free boundary of the collar of $\mathcal{S}$. No already constructed approach path intersects $\mathcal{S}$.

The distinguished edge h_{APA} is called the *approach path aperture* of $\mathcal{S}$. The existence of the line segment, refered to in condition AP2, will allow us to extend the approach path when visiting new triangles.

Lemma 6. *The main procedure* ConnectedSum *can be enhanced in such a way that:*
1. It maintains the invariants (AP1), (AP2) and (AP3)
2. If visiting a triangle t causes a splitting extension such that $\mathcal{S} \setminus t$ is connected (Figure 5, line 5–7), it constructs a pair of generators in $O(n)$ time.

Proof. Before describing the actual enhancement of ConnectedSum, we impose some restrictions on the traversal and the approach paths, and introduce some primitive operations that facilitate the description of the algorithm.

We require that, during the traversal of the surface, the next triangle visited in a call of ConnectedSum on $\mathcal{S}$ is incident upon the approach path aperture h_{APA}, contained in the boundary $\mathcal{B}$ of $\mathcal{S}$. Furthermore, we require that approach paths do not intersect vertices of $\mathcal{M}$.

A basic operation is that of *cloning an approach path*. Cloning an approach path γ_{AP}, directed from p_0 to its terminal vertex on the approach path aperture h_{APA}, amounts to constructing a PL-path from p_0 to h_{APA}, with the same combinatorial structure as γ_{AP} (i.e., intersecting the same sequence of edges and faces of $\mathcal{M}$). This clone should not share any point with already constructed approach paths or generators, apart from p_0. To avoid ambiguities, we assume that a clone runs to the left of its original. In view of condition (AP1), any approach path can be cloned, and cloning can even be repeated on clones.

Furthermore, we employ the notion of *routing a PL-curve* along (part of) the free boundary of a PL-collar. This operation is similar to cloning, in that we construct a PL-curve inside the PL-collar, which has the same combinatorial structure as (a sub-path of) the free boundary of the collar. We require this curve to be disjoint from already constructed generators and approach paths, which is possible in view of conditions (AP1) and (AP3).

Now consider a *regular extension*. Set the approach path aperture h'_{APA} of $\mathcal{S}' = \mathcal{S} \setminus \{t\}$ to one of the half-edges in the boundary of t, not incident upon $\mathcal{B}$ (e.g. h_3 in Figure 4). According to (AP2), there is a line segment $s = pp'$ connecting the terminal vertex p of γ_{AP} with a point p' inside t and on the free boundary of the collar of $\mathcal{S}$. Let q be a point on h'_{APA} not belonging to the collar of $\mathcal{S}$. Such a point exists, since h'_{APA} does not belong to $\mathcal{B}$, and since the PL-collar of $\mathcal{S}$ only intersects faces incident upon $\mathcal{B}$. Extending γ_{AP} with pp' and $p'q$, we obtain an approach path γ'_{AP} for $\mathcal{S}'$ satisfying (AP1). Furthermore, since q does not belong to the collar of $\mathcal{S}$ there is a line segment qq', with q' on the free boundary of the collar of $\mathcal{S}'$, that is disjoint from the collar of $\mathcal{S}$. In other words, (AP2) holds for $\mathcal{S}'$. Since we do not complete any generators, (AP3) also holds for $\mathcal{S}'$. The enhanced version of a regular extension obviously takes $O(1)$ time. It remains to consider a *splitting extension*. If $\mathcal{S}' = \mathcal{S} \setminus \{t\}$ is disconnected, and both g_l and g_r are positive, we construct a clone γ'_{AP} of the approach path γ_{AP}. Now we extend γ'_{AP} to the half-edge v_1v_s of $\mathcal{B}_l$, and we extend γ_{AP} to the half-edge v_sv_2 of $\mathcal{B}_r$ (the notation is as in Section 2); See Figure 7, Top. Arguing as in the case of a regular extension, we conclude that conditions (AP1), (AP2) and (AP3) hold for the connected components $\mathcal{S}_l$ and $\mathcal{S}_r$ of $\mathcal{S}'$, with approach path apertures v_1v_s and v_sv_2, respectively. If g_l or g_r is zero, we just extend the approach path to the non-visited part of positive genus in $O(1)$ time. Cloning only needs to be done in case the genus of both non-visited parts is less than the genus of $\mathcal{S}$, which happens at most $g-1$ times. Therefore, the overall complexity of all splitting extensions of this type is $O(gn)$.

Finally, consider a splitting extension in which $\mathcal{S}' = \mathcal{S} \setminus \{t\}$ is connected. Now we construct *four disjoint clones* γ_1, γ_2, $\overline{\gamma}_1$ and $\overline{\gamma}_2$ of γ_{AP}, whose respective end-points p_1, p_2, $\overline{p}_1$ and $\overline{p}_2$, occur in this order on the approach path aperture h_{APA} between v_1 and the end-point of γ_{AP}. The approach path γ_{AP} is now extended to the half-edge v_sv_2, see Figure 7, Bottom. As before, we can do this in such a way that (AP1), (AP2) and (AP3) holds for $\mathcal{S}'$. Finally, we complete a pair of generators by connecting the end-points of γ_1 and γ_2 with the end-points of $\overline{\gamma_1}$ and $\overline{\gamma_2}$, respectively, by two curves σ_1 and σ_2; See Figure 7, Bottom. More precisely, let $\mathcal{F}$ and $\mathcal{F}'$ be the free parts of the collars of $\mathcal{S}$ and $\mathcal{S}'$. Then σ_1 is a PL-curve obtained by connecting p_1 to a point near v_1 on v_1v_s by a curve inside t, and subsequently routing

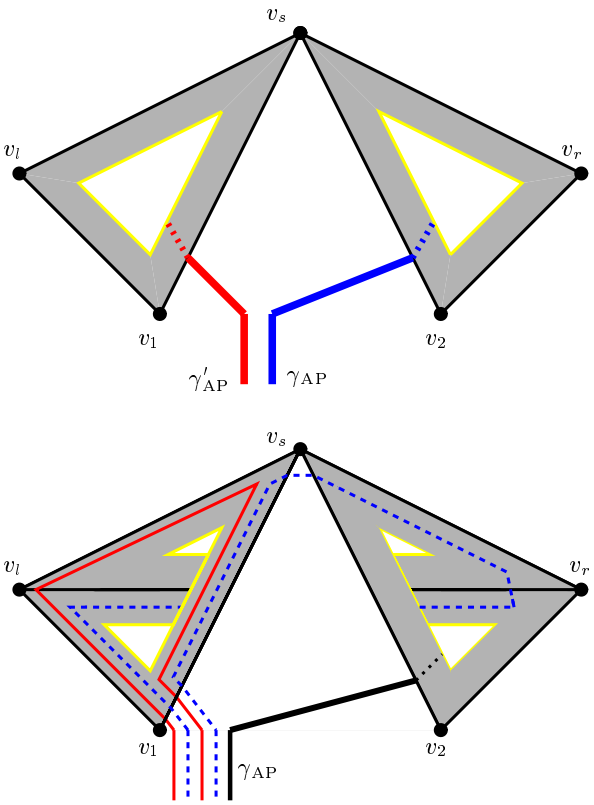

Figure 7: Splitting extensions upon visit of triangle $t = v_1v_2v_3$. Top: $\mathcal{S} \setminus \{t\}$ is not connected. Bottom: $\mathcal{S} \setminus \{t\}$ is connected, so a pair of generators is constructed.

it along the part of $\mathcal{F}$ near $v_1 \overset{*}{\to} v_l \overset{*}{\to} v_s$ and along the part of $\mathcal{F}'$ near $v_s \overset{*}{\to} v_1$, and, finally, connecting it to $\overline{p_1}$. Furthermore, σ_2 is a PL-curve obtained by connecting p_2 to a point near v_1 on v_1v_s, and subsequently routing it along the part of $\mathcal{F}'$ near $pv_1 \overset{*}{\to} v_l \overset{*}{\to} v_r$, then along the part of $\mathcal{F}$ near $v_l \overset{*}{\to} v_s$, letting it traverse t near v_s, then routing it along the part of $\mathcal{F}'$ near $v_s \overset{*}{\to} v_1$, and, finally, connecting it to $\overline{p}_2$ by a curve inside t. Obviously, σ_1 and σ_2 do not intersect the free boundary of the collar of $\mathcal{S}'$. Furthermore, it is easy to see that these curves can be constructed in such a way that they are disjoint from any generators or approach paths already constructed.

The time complexity of this splitting operation is $O(n)$, since the generators share only a constant number of edges and vertices with each edge and face of $\mathcal{M}$. Since there are exactly g splitting extensions of this type, the overall time complexity is $O(gn)$. □

Observe that ConnectedSum constructs pairs of generators that are not interleaved near the base point. In other words, the g pairs of generators form a *canonical* set. Theorem 1 is now a straightforward consequence of Corollary 5 and Lemma 6.

Remark: connected sums. In fact, ConnectedSum rewrites the initial surface $\mathcal{M}$ as a connected sum (cf [1, Chapter 7]) of g tori. To see this, let $\overline{\mathcal{S}}$ denote the closed orientable surface obtained by gluing a disk along the boundary of a collared surface $\mathcal{S}$.

The algorithm maintains the surface $\mathcal{M}$ as a connected sum of the form $\mathcal{M} = \overline{\mathcal{S}_1} \sharp \cdots \sharp \overline{\mathcal{S}_k}$, where $\mathcal{S}_i$ is a collared surface, $g_i := \mathrm{genus}(\mathcal{S}_i) > 0$, and $\sum_{i=1}^{k} g_i = g$. Here $\sharp$ denotes the connected sum-operator. Initially, $k = 1$ and $\mathcal{S}_1 = \mathcal{M} \setminus t_0$. Consider a recursive call of ConnectedSum on $\mathcal{S} = \mathcal{S}_i$. If a pair of generators is constructed (line 7), $\overline{\mathcal{S}}$ is rewritten as $\overline{\mathcal{S}} = \overline{\mathcal{S} \setminus (t \cup \gamma)} \sharp \mathbb{T}^2$, where $\mathbb{T}^2$ is a 2-torus. If, on the other hand, $\mathcal{S} \setminus t$ is disconnected, $\overline{\mathcal{S}}$ is rewritten as $\overline{\mathcal{S}} = \overline{\mathcal{S}_l} \sharp \overline{\mathcal{S}_r}$. In the latter case, we only make progress if neither $\overline{\mathcal{S}_l}$ nor $\overline{\mathcal{S}_r}$ are topological disks. This corresponds to the subtlety discussed at the end of Section 3.

5. BRAHANA'S ALGORITHM

The inverse of a path p is denoted by $\imath(p)$ or p^{-1}, and for a set of paths S we denote the set $S \cup \imath(S)$ by $\hat{S}$.

Let G be a maximal subgraph of the vertex-edge graph of $\mathcal{M}$ such that $\mathcal{M} \setminus G$ is connected, and let T_G be a tubular neighbourhood of G in $\mathcal{M}$. By construction, $\mathcal{M} \setminus T_G$ is a topological disk and G is a deformation retract of T_G. Therefore a set of generators of the fundamental group $\pi(G, x)$ of G at x is also a set of generators of the fundamental group $\pi(\mathcal{M}, x)$ of $\mathcal{M}$ at x. We can decompose our method into three steps:

1. First we construct a set $\mathcal{G}$ of $(2g)$ generators of $\pi(G, x)$, associated with a set E of $(2g)$ directed edges of $\mathcal{M}$ under a bijection $\ell : E \to \mathcal{G}$, and a cycle ϕ of $\hat{E}$ such that for $e \in \hat{E}$:

$$\ell(\phi(e))\ell(\phi^2(e)) \ldots \ell(\phi^{4g}(e)) \sim \epsilon_x \qquad (*)$$

in $\pi(\mathcal{M}, x)$. Here ϵ_x is the trivial path at x.

2. Secondly, we transform in $O(gn)$ time the set $\mathcal{G}$ into a set $\mathcal{H}$ of generators x_i, y_i of $\pi(G, x)$, each of linear complexity, such that a loop in $\mathcal{H}$ is homotopic (in G) to the concatenation of $O(g)$ loops in $\hat{\mathcal{G}}$, and the relation satisfied by the x_i, y_i in $\pi(\mathcal{M}, x)$ is in 'canonical form', i.e.

$$[x_1, y_1] \cdots [x_g, y_g] \sim \epsilon_x. \qquad (**)$$

As usual, $[x_i, y_i]$ is the commutator $x_i y_i x_i^{-1} y_i^{-1}$, and $\sim$ denotes path-homotopy.

3. Finally, we show how to construct in $O(gn)$ time a canonical set of generators x_i^*, y_i^* of $\pi(\mathcal{M}, x)$ such that $x_i \sim x_i^*$ and $y_i \sim y_i^*$ in T_G.

Step 1. We construct a spanning tree T of G rooted at x. Let E denote the set of non-tree edges in G; Each edge in E is oriented arbitrarily and each edge in T is oriented towards the root. Without loss of generality we assume for convenience that there is only one edge e_{sink} of the tree incident upon x. For each (directed) edge $e \in \hat{E}$ we consider the shortest edge-path $\gamma_e = ee_1e_2 \cdots e_{\mathrm{sink}}$ from e to x in T. By construction, for $e \neq e'$ the paths γ_e and $\gamma_{e'}$ coincide only on a proper suffix sub-path, i.e., both paths can be decomposed as $\gamma_e = \tau_{e,e'}\gamma_{e,e'}$ and $\gamma_{e'} = \tau_{e',e}\gamma_{e',e}$, where $\gamma_{e,e'} = \gamma_{e',e}$ and $\tau_{e,e'}$ and $\tau_{e',e}$ are disjoint except at their sink $v(e, e')$. One can check that the relation on the edges in $\hat{E}$ defined by $e \prec e'$ if the sink edges of $\tau_{e,e'}$, $\tau_{e',e}$ and the source edge of $\gamma_{e,e'}$ are in counterclockwise order around their common endpoint $v(e, e')$

— with respect to the choice of an orientation of the surface $\mathcal{M}$ — is a transitive relation.

Let now $\ell(e)$ be the loop with basepoint x obtained by concatenation of the loops $\imath(\gamma_e)$ and $\gamma_{\imath(e)}$, removing one of the two occurrences of e^{-1}, i.e., $\ell(e) = \imath(\gamma_e)\gamma_{\imath(e)}$. Note that $\ell(\imath(e)) = \imath(\ell(e))$. The set $\mathcal{G} := \ell(E)$ is a set of $(2g)$ generators of $\pi(G, x)$, and consequently of $\pi(\mathcal{M}, x)$. Furthermore, the unique relation in $\pi(\mathcal{M}, x)$ satisfied by these generators is $(*)$, where the operator ϕ is defined by $\phi(e) = \psi \circ \imath(e)$. Here $\psi(e)$ is the successor of e with respect to the circular order on $\hat{E}$, induced by the linear order $\prec$.

Step 2. We use a sequence of Brahana transformations, cf [10]. Let $\ell_i = \ell(\phi^i(e))$ for some $e \in \hat{E}$, and let M be the loop $\ell_1 \cdots \ell_{4g}$. The loop M can be decomposed into $aX_1bX_2a^{-1}X_3b^{-1}X_4$, where a and b are loops in $\hat{\mathcal{G}}$, and X_4 is nonempty (unless X_1, X_2 and X_3 are empty, in which case we are done). If X_1, X_2, X_3 are not all empty we replace the loops a and b by the loops $x = aX_1bX_2a^{-1}$ (consequently $b^{-1} = X_2a^{-1}x^{-1}aX_1$) and $y = X_3X_2a^{-1}$ ($a = y^{-1}X_3X_2$) to obtain successively

$$M \sim \overbrace{aX_1bX_2a^{-1}}^{x} X_3b^{-1}X_4 \sim x \overbrace{X_3X_2a^{-1}}^{y} x^{-1}aX_1X_4$$
$$\sim [x, y]X_3X_2X_1X_4 \sim X_3X_2X_1X_4[x, y].$$

If X_1, X_2, X_3 are all empty, then we simply set $x = a$, $y = b$. In both cases $M \sim M'[x, y]$ where M' is the concatenation in some order of the loops in $\hat{\mathcal{G}} \setminus \{a, a^{-1}, b, b^{-1}\}$, and where x and y are loops composed of $O(g)$ generators in $\hat{\mathcal{G}}$. The loops a and b and their corresponding edges in $\hat{E}$ are said to be *converted*. After j such transformations we have converted a set $\mathcal{G}_j$ of $2j$ generators in $\mathcal{G}$ into a set $\mathcal{H}_j$ of $2j$ generators $x_1, y_1, x_2, y_2, \ldots, x_j, y_j$, such that $M \sim M_j[x_1, y_1] \cdots [x_j, y_j]$. Here M_j is the concatenation in some order of the loops in $\hat{\mathcal{G}} \setminus \hat{\mathcal{G}}_j$. For $j = g$ we obtain generators which satisfy $(**)$, but whose total complexity is only in $O(g^2 n)$.)

We now explain how to reduce the complexity of these loops by homotopy to $O(gn)$. First we examine how the relation $\phi = \psi \circ \imath$ is tranformed. For $j \geq 0$ and for $e \in \hat{E} \setminus \hat{E}_j$ we define $\psi_j(e)$ to be the $\prec$-successor of e in $\hat{E} \setminus \hat{E}_j$, and $\phi_j(e)$ to be the edge e' such that the successor of $\ell(e)$ in M_j is $\ell(e')$.

LEMMA 7. *$\phi_j(e) = \psi_j \circ \imath(e)$.*

PROOF. We prove the result by induction. The case $j = 0$ follows from the definition of ϕ. Let a and b be the loops converted at step $j + 1$. One has $M_j = aX_1bX_2a^{-1}X_3b^{-1}X_4$ and $M_{j+1} = X_3X_2X_1X_4$. Let $e' = \phi_{j+1}(e)$. If $e' = \phi_j(e)$, then $e' = \psi_j(\imath(e)) = \psi_{j+1}(\imath(e))$, since e and e' are not converted at step $j+1$. Assume now that $e' \neq \phi_j(e)$, and let e_i^k for $k = 1, 2$ and $i = 1, 2, 3, 4$ be defined by $X_i = \ell(e_i^1)X_i'\ell(e_i^2)$ if X_i is non empty. The pair (e, e') coincides with one of the pairs $(e_k^2, e_{k'}^1)$ where k precedes k' in the order 3,2,1,4. For example if $e = e_3^2$ and $e' = e_2^1$ then $\psi_j(\imath(e_3^2)) = \phi_j(e_3^2) = \imath(b)$ and $\psi_j(\imath(b)) = \phi_j(b) = e_2^1 = e'$. Therefore, $\psi_{j+1}(\imath(e)) = e'$. The other cases are similar. □

We are now ready to decrease in optimal time the complexity of the loops x_i, y_i. Assume that

$$x_j = \ell(a)\ell(b)\cdots\ell(z)$$

and let $sc(x_j)$ be the loop defined by

$$\imath(\gamma_a)\ell(\imath(a), b)\ell(\imath(b), c)\cdots\ell(\imath(y), z)\gamma_{\imath(z)},$$

where $\ell(e, e')$ is the concatenation of the two paths $\tau_{e,e'}$ and $\imath(\tau_{e',e})$. Clearly $x \sim sc(x)$, and the size of $sc(x)$, i.e., its number of edges in $\mathcal{M}$, is in $O(n)$. Starting from its source e, we can visit the edges of $\ell(e, e')$ in time proportional to its size if we can determine efficiently the vertex $v(e, e')$. In view of Lemma 7 this can easily be done in $O(1)$ time, provided we maintain for each node v of the tree T the $\prec$-ordered list $L_j(v)$ of edges $e \in \hat{E}$ whose corresponding loops have non yet been converted, and whose associated paths γ_e lie along v. The lists $L_0(v)$ are easily created in $O(gn)$ time, and updated in $O(n)$ time, each time an edge is converted by a traversal of the corresponding loop.

Step 3. Omitted from this version.

6. IMPLEMENTATION

We have implemented both the incremental and Brahana's algorithm in `C++`, using the CGAL [7][1] polyhedron data structure. Both source codes are approximately 3,000 lines long. The remaining issue in the implementation is the representation of loops. In practice, a PL-loop is specified by the list of edges it crosses. Also, each edge of the combinatorial surface points to the list of loops it is crossed by. See Figure 8. In order

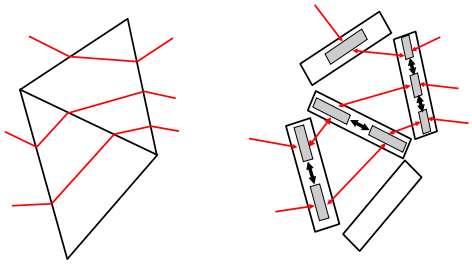

Figure 8: The loops data structure.

to visualize the PL-loops, we uniformly insert in each edge a number of points equals to the size of its list. We then link these points according to each loop list.

In Section 4 we always visit a triangle incident upon the approach path aperture. In practice, we can choose any triangle incident to the boundary and keep the same complexity. In our implementation we use a 'potato peeling' traversal. This heuristic produces nicer loops.

We run our programs on a test suite of threads of tori with various lengths (Figures 9 and 10).

Figure 11 shows the genus 3 torus and its system of loops computed by each method. Results for the test suite are reported Table 1. Times were obtained with the `CGAL::Real_timer` class. For both algorithms times

[1] `http://www.cgal.org/`

Table 1: Statistics for the incremental and Brahana's algorithms

Object	genus	#faces	time for reading in sec.	incremental method		Brahana's method	
				time in sec.	total size of loops	time in sec.	total size of loops
torus_10	10	408	0.006	0.014	2,687	0.015	2,232
torus_20	20	808	0.01	0.04	9,472	0.04	7,527
torus_50	50	2,008	0.03	0.16	53,827	0.24	40,812
torus_100	100	4,008	0.06	0.52	207,752	1.17	154,287
torus_200	200	8,008	0.12	1.93	815,602	6.71	598,737
torus_500	500	20,008	0.43	213.3	5,039,152	-	-
torus_3	3	128	0.002	0.004	320	0.003	255
torus_3_1	3	384	0.006	0.01	542	0.009	673
torus_3_2	3	1,152	0.02	0.03	1,168	0.03	1,144
torus_3_3	3	3,456	0.06	0.09	2,064	0.1	1,906

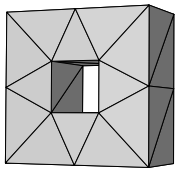

Figure 9: All tori were obtained by gluing translated copies of this genus 1 torus.

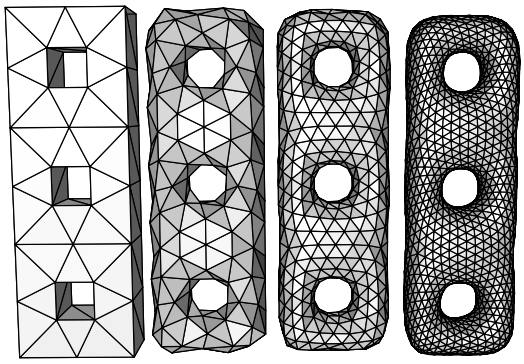

Figure 10: A genus 3 torus with various subdivisions.

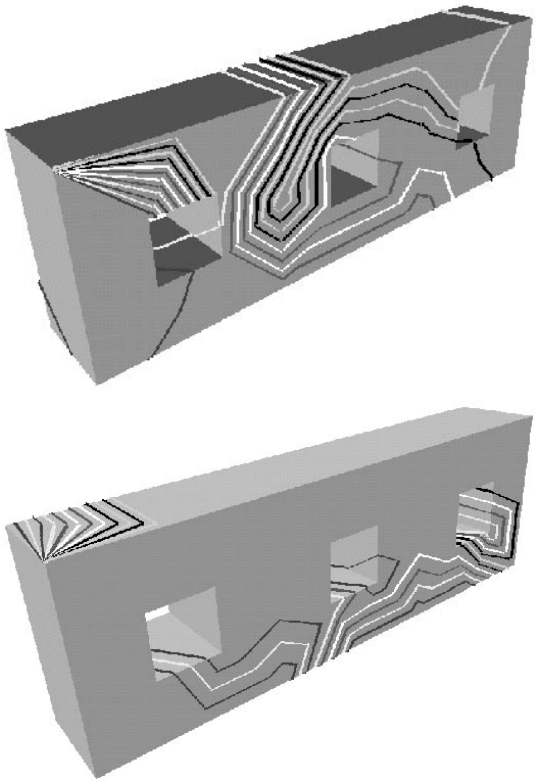

Figure 11: Results of the incremental (top) and Brahana's (bottom) methods for a genus 3 torus.

include reading input files, computing the canonical system of loops with the data structure mentioned above (with the uniform embedding). They do not include, however, time for writing the resulting PL-loops into a file. Tests were all run on a Pentium III, 800 MHz with 256 Mb RAM. For all the tori we used base-point with index 0 near one extremity of the threads. Note that the tori present a worst case configuration (See also Figure 2). As asserted by Theorem 1, time is roughly proportional to the product gn of the genus by the surface complexity. This is also the case for the total size of the loops. For the torus with genus 500 and up timings are not representative as a large amount of memory swapping is involved. Table 2 shows the influence of the choice of the base-point on the torus_10 example. Points 0 and 184 ly approximately at the two extremities of the thread while point 92 is in the middle. A closer look at the execution profile of Brahana's method shows that most of the time is spent at transforming the initial system of loops into a canonical one. This holds particularly when the initial set of generators satisfies a relation 'close' to the other canonical form: $a_1 b_1 a_2 b_2 \dots a_g b_g \overline{a_1}\,\overline{b_1} \dots \overline{a_g}\,\overline{b_g}$. In this case, the final generators are indeed expressed as $\Omega(g^2)$ initial generators. The construction of the maximal subgraph G and the choice of the base-point have great influence on the re-

Table 2: Influence of the base-point

base-point index	Size of loops for incremental	Size of loops for Brahana
0	2687	2232
2	2060	4693
91	2177	2004
92	1611	4334
93	2248	3409
100	1631	4157
150	2190	5176
184	2380	6725

sults. In order to construct G we use a spanning tree on the dual graph of faces. The edges in the dual graph are in 1-1 correspondence with the edges of the surface. G is composed of the surface edges not in the spanning tree. We have tested two traversals to construct the dual spanning tree: a peeling traversal - a specific depth-first search - and a breadth-first search. Figure 13 illustrates the resulting loops on a simple torus example. Breadth-first search always gave much shorter loops on our examples. This figure and the following Figures 14 and 15 were obtained by running our algorithm on the combinatorial surfaces with the subdivision shown in Figure 12 after identification of the boundary sides in a canonical order. However, as on this Figure, we kept a planar embedding for the vertex coordinates, faking a flattening of the cut open surfaces. These flattenings obviously correspond to particular canonical schemata which are a priori not related to the computed canonical system of generators.

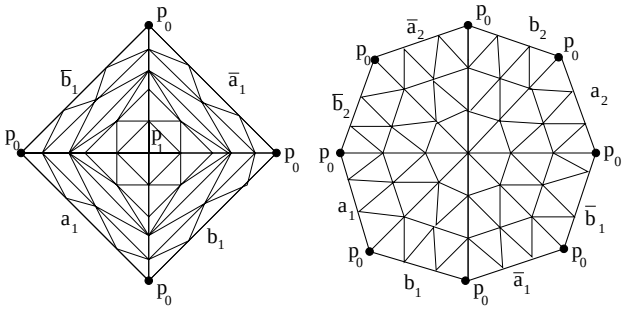

Figure 12: Left: torus subdivision. Right: double torus subdivision.

We comment on Brahana's algorithm applied to the double torus example on the left Figure 12 with base point p_0.

The initial loop M (beginning of step 2 in Brahana's algorithm) turns out to be:

$$(+0)(-3)(-1)(+2)(-0)(+1)(-2)(+3)$$

In the notation of Section 5, the algorithm takes $a = (+0)$ and $b = (-3)$. The first part of the Brahana transformation is:

$$x = aX_1bX_2a^{-1} = (+0)(-3)(-1)(+2)(-0)$$

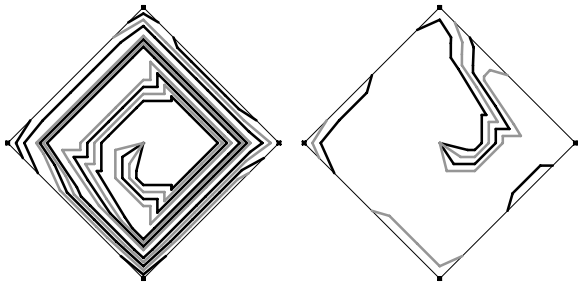

Figure 13: The shown loops, dark and light grey, were computed with Brahana's method and center base point p_1 using a peeling traversal (left) and a breadth-first traversal (right).

so $X_1 = \epsilon$, $X_2 = (-1)(+2)$, $X_3 = (+1)(-2)$ and $X_4 = \epsilon$ Thus M becomes:

$$(+3)(+1)(-2)(-1)(+2)(-0)(-3)(+0)$$

The second part of this Brahana transformation corresponds to the choice
$y = X_3X_2a^{-1} = (+1)(-2)(-1)(+2)(-0)$, and M becomes

$$(+x)(+y)(-x)(-y)(+1)(-2)(-1)(+2)$$

Figure 14 shows the initial loops, while Figure 15 shows the converted canonical set of loops. In order to help visualization, each pair of generators is first drawn separately.

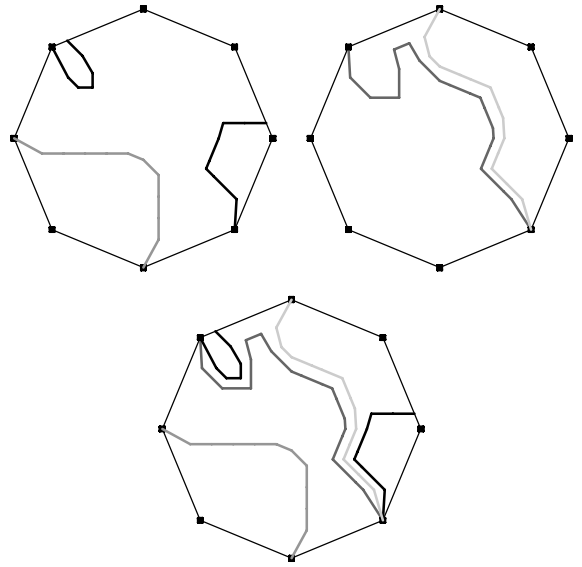

Figure 14: Upper left: Pair (0,3) of initial loops. Upper right: Pair (1,2). Bottom: The initial set of loops.

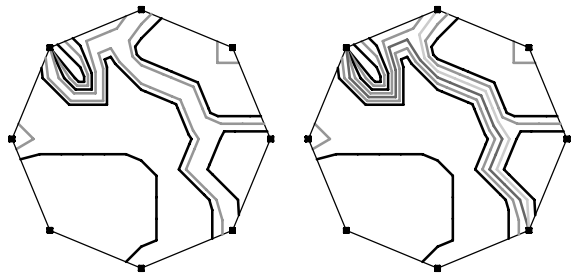

Figure 15: Left: pair (x, y) replaces pair (0,3). Right : the canonical set of loops (pair (1,2) is left unchanged).

7. FINAL REMARKS

We have presented two algorithms for computing a canonical set of PL-generators on an orientable triangulated surface. Both algorithms are worst-case optimal. Note that Brahana's algorithm can actually be applied to any combinatorial surface, not necessarily triangulated. It seems that this algorithm could also be applied to non orientable surfaces but loosing the $O(gn)$ complexity. It is not clear whether one of the two algorithms generally produces better results in terms of the loops complexity. In any case the obtained PL-loops look much too jaggy and complex to be of any use for practicle applications such as morphing. More work needs to be done in this direction taking into account the "geometry" of the surface.

8. REFERENCES

[1] M. Armstrong. *Basic Topology*. Undergraduate Texts in Mathematics. Springer Verlag, Berlin, 1983.

[2] T. Brahana. Systems of circuits on 2-dimensional manifolds. *Ann. Math.*, 23:144–168, 1921.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[4] Dey and Guha. Transforming curves on surfaces. *JCSS: Journal of Computer and System Sciences*, 58:297–325, 1999.

[5] T. Dey, H. Edelsbrunner, and S. Guha. Computational topology. In *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 109–143, 1999.

[6] T. Dey and H. Schipper. A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection. *Discrete and Computational Geometry*, 14:93–110, 1995.

[7] L. Kettner. Using generic programming for designing a data structure for polyhedral surfaces. *Computational Geometry: Theory and Applications*, 13:65–90, 1999.

[8] J. Stillwell. *Classical Topology and Combinatorial Group Theory*. Springer-Verlag, New York, 1993.

[9] G. Vegter. Computational topology. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 28, pages 517–536. CRC Press LLC, Boca Raton, FL, 1997.

[10] G. Vegter and C. K. Yap. Computational complexity of combinatorial surfaces. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 102–111, 1990.