TECHNISCHE UNIVERSITÄT CHEMNITZ

Maharavo Randrianarivony Guido Brunnett

$\mathcal{C}^0$-paving of closed meshes with quadrilateral patches

Preprint SFB393/05-17

NUMERISCHE PARALLELE SIMULATION SFB 393

Sonderforschungsbereich 393

Parallele Numerische Simulation für Physik und Kontinuumsmechanik

TECHNISCHE UNIVERSITÄT CHEMNITZ

Sonderforschungsbereich 393

Parallele Numerische Simulation für Physik und Kontinuumsmechanik

Maharavo Randrianarivony Guido Brunnett

$\mathcal{C}^0$-paving of closed meshes with quadrilateral patches

Preprint SFB393/05-17

Preprintreihe des Chemnitzer SFB 393

ISSN 1619-7178 (Print) ISSN 1619-7186 (Internet)

SFB393/05-17 December 2005

Contents

Authors' addresses:

Maharavo Randrianarivony, Guido Brunnett
TU Chemnitz, Fakultät für Informatik
D-09107 Chemnitz
`http://www.tu-chemnitz.de/~ranh`
`http://www.informatik.tu-chemnitz.de/~gdv`

Abstract

We would like to decompose a closed orientable 2-manifold with arbitrary genus g into a set of four-sided tessellants. The input surface is a triangular surface mesh representing a free-form shape. The search for canonical curves is done in two steps: finding initial curves by means of algebraic or geometric methods and improving them by using combinatorial optimization. Afterward, we perform the handle decomposition by slicing the surface along those curves. After flattening the surface on the plane by using the setting $(a_1 b_1 a_1^{-1} b_1^{-1}) \cdots (a_g b_g a_g^{-1} b_g^{-1})$, we use the resulting parametrization to split the surface into four-sided patches. We approximate those patches by simple surfaces like Bézier in which we want to obtain global $\mathcal{C}^0$-joints. The proposed methods will be illustrated by several numerical examples.

1 Introduction

It often happens in practice that we have a numerical solver of integral equations which accepts mesh-free geometric data as input but the only geometry that we have at our disposal is a mesh. Treating every element of the mesh as a patch will surely explode the computational costs of the integral solvers. Faced by such a conflicting situation, we have only one choice if we do not want to reject the input mesh: we try to generate a few patches from the mesh. That process amounts to approximating or interpolating the mesh by practical surfaces such as Bézier patches (Fig. 1) and that is exactly the purpose of this paper where we will treat only free-form surfaces. For algebraic surfaces such as spheres, planes, cylinders or combinations of them, we recommend [22, 21]. We will also assume that our surface is orientable and closed. Those types of surfaces already cover a lot of interesting practical cases.

Apart from the fitting process, two main difficulties have to be considered. First, the partitioning of the large mesh into pieces which can be approximated by four-sided patches. Second, having a parameter 2D meshes which allow us to perform the approximation by parametric surfaces.

Let us suppose that we have a triangular mesh $\mathcal{M}$ which represents a surface embedded in the space. Our primary concern in this paper is to present a methodology to decompose a closed surface mesh if arbitrary genus into a set of four-sided pieces. The approximation of an individual four-sided region by a continuous patch is only briefly described in section 7. The main idea of triangulation paving is processed in four main steps. First, we have to find some curves such that if the surface $\mathcal{M}$ is split along these curves, we still have a single connected surface. The number of these curves depends on the genus of the surface $\mathcal{M}$. Second, we want to have a topological parameterization from a polygonal disk $P \subset \mathbf{R}^2$, which we still have to determine, to the mesh in which we have to take the topological properties of the mesh into account. The third step consists in splitting the parameter domain into four-sided subregions. The last step consists in fitting a surface from each four-sided quadrilateral of the polygonal disk P to the corresponding mesh portion. During the fitting process, we require C^0 joint. In the next sections, we would like to describe those steps one by one.

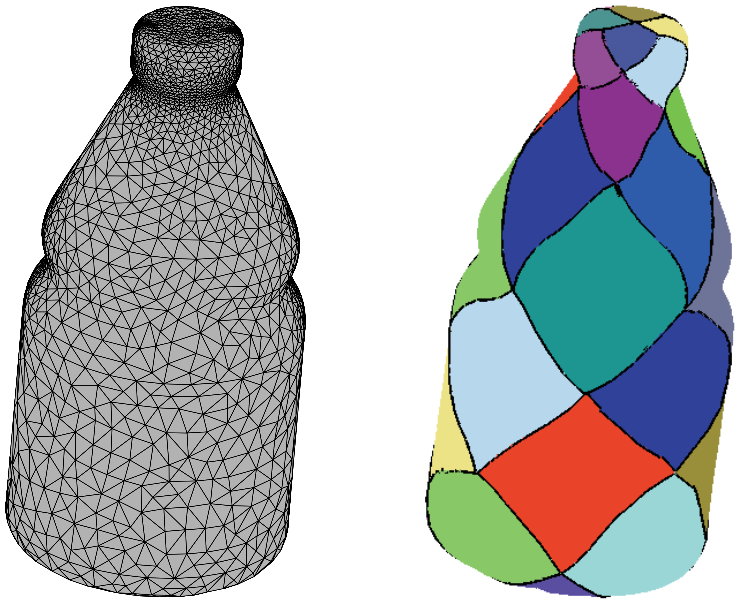


Figure 1: Triangulation paving: (a) bottle in form of a triangular mesh (b)approximated surface in form of patches.

2 Handle decomposition

Suppose we have a closed surface $\mathcal{M}$ which is orientable and of genus $g > ing0$. We define a system of canonical curves to be a set of $2g$ closed curves $\mathcal{A}_1$, $\mathcal{B}_1$,...,$\mathcal{A}_g$, $\mathcal{B}_g$ which fulfills the following criteria.

(C1) They all reside on the surface $\mathcal{M}$.

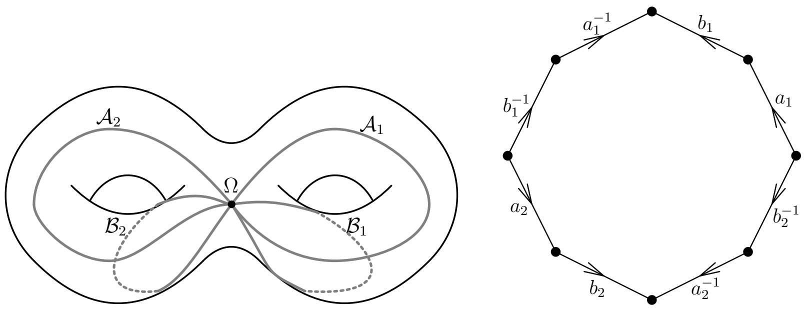
(C2) They have one and only one intersection Ω known as basepoint.

(C3) If we cut the surface $\mathcal{M}$ along those curves then we still have a connected surface (Fig. 2) which can be flatten out to become a planar polygon [3, 7].

$$a_1 b_1 a_1^{-1} b_1^{-1} ... a_g b_g a_g^{-1} b_g^{-1}. \tag{1}$$

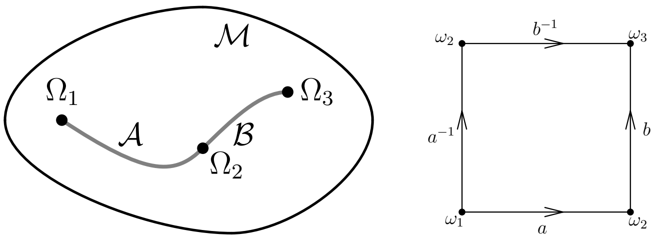
The planar edges a_i (resp. b_i) correspond to the canonical curves $\mathcal{A}_i$ (resp. $\mathcal{B}_i$). The exponent -1 in (1) specifies that the corresponding edge is to be traversed in the opposite direction.

The first step in our algorithm is to search for those canonical curves in which achieving criterion (C2) is the most difficult task. The process of cutting the surface $\mathcal{M}$ along these curves is better known as handle decomposition. After finding the canonical curves, we will describe an approach to define a parameterization from a planar polygon to the surface.



(a) (b)

Figure 2: (a) A surface with genus 2 and its four canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$, $\mathcal{A}_2$, $\mathcal{B}_2$ (b)The flattened planar polygon $a_1 b_1 a_1^{-1} b_1^{-1} a_2 b_2 a_2^{-1} b_2^{-1}$



(a) (b)

Figure 3: Flattening: (a) Genus zero surface (b) Its flattened planar polygon

Remark 1 For surfaces of genus zero, the problem of splitting is much simpler because we need only to split the mesh along two abutting curves $\mathcal{A}$ and $\mathcal{B}$ as in Fig. 3(a). Those 3D-curves will be flattened in the plane to obtain the boundary 2D-curves a, b, a^{-1}, b^{-1} as in Fig. 3(b).

3 Preliminary search for canonical curves

Searching for the canonical curves is usually done in two stages. First, one finds some preliminary canonical curves $\overline{\mathcal{A}}_1$, $\overline{\mathcal{B}}_1$,...,$\overline{\mathcal{A}}_g$,$\overline{\mathcal{B}}_g$ which fulfill the above criteria (C1), (C2), (C3) but which are undesirable in practice. The second stage consists of improving those preliminary curves by shifting them homotopically in order to find the final canonical curves $\mathcal{A}_1$,$\mathcal{B}_1$,...,$\mathcal{A}_g$,$\mathcal{B}_g$. In this section, we would like to discuss how to achieve the first stage with two different methods. The first one is based on algebraic operations and the second one is a combinatorial approach.

3.1 Smith normal form and normalized canonical form

Let us denote by n_t, n_e, and n_v the number of triangles , edges and vertices respectively of the given mesh $\mathcal{M}$. For topological terms which are not clear , refer to the appendices or [11, 15]. We will denote the i-th triangle, the j-th edge and the k-th vertex of the mesh $\mathcal{M}$ by σ_i^2, σ_j^1, σ_k^0 respectively. We would like to describe briefly the way we compute the homology bases which represent some curves drawn on the surface (compare with Fig. 4(a)).
Consider the two incidence matrices [9] E_0 and E_1:

$$E_0 := \begin{bmatrix} [\sigma_1^1 : \sigma_1^0] & ... & [\sigma_{n_e}^1 : \sigma_1^0] \\ ... & ... & ... \\ [\sigma_1^1 : \sigma_{n_v}^0] & ... & [\sigma_{n_e}^1 : \sigma_{n_v}^0] \end{bmatrix} \quad E_1 := \begin{bmatrix} [\sigma_1^2 : \sigma_1^1] & ... & [\sigma_{n_f}^2 : \sigma_1^1] \\ ... & ... & ... \\ [\sigma_1^2 : \sigma_{n_e}^1] & ... & [\sigma_{n_f}^2 : \sigma_{n_e}^1] \end{bmatrix} \tag{2}$$

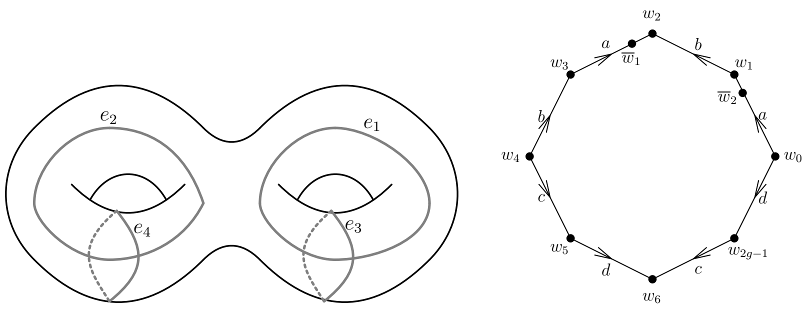
where $[\sigma_1^s : \sigma_1^{s-1}]$ denotes the incidence number of σ_1^s and σ_1^{s-1}.
Since those incidence numbers take integer values, we have to deal with integer computations. That is, we may not perform any division operations. We have effectively the following relation pertaining to the operators E_0 and E_1:

$$\mathbf{Z}^{n_f} \xrightarrow{E_1} \mathbf{Z}^{n_e} \xrightarrow{E_0} \mathbf{Z}^{n_v}. \tag{3}$$

The property of the boundary operator [20] implies in particular

$$E_0 \cdot E_1 = 0\,. \tag{4}$$

The set σ_i^k forms now a basis of the k-chains C_k. The reduction into normalized canonical form consists in searching for new bases of C_0, C_1, C_2 such that in those bases the above



(a) (b)

Figure 4: (a)Homology bases e_1, e_2, e_3, e_4 (b)Coincidence of the images of different parameter nodes

matrices take the form

$$\tilde{E}_0 = \begin{bmatrix} 0 & \vdots & \Lambda_0 \\ \cdots & . & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \qquad \tilde{E}_1 = \begin{bmatrix} 0 & \vdots & \Lambda_1 \\ \cdots & . & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \tag{5}$$

where Λ_0 and Λ_1 are square diagonal matrices. In other words, we are searching for three square matrices M_v, M_e, and M_t of size n_v, n_e, and n_f respectively such that

$$\tilde{E}_1 = M_e^{-1} E_1 M_t \qquad \tilde{E}_0 = M_v^{-1} E_0 M_e. \tag{6}$$

The new bases of C_0, C_1 and C_2 are therefore the columns of the matrices M_v, M_e, and M_t. Denote by γ_0 and γ_1 the dimensions of the square diagonal matrices Γ_0, Γ_1. Let us consider the new basis $\{e_1, ..., e_{n_e}\}$ which is provided by M_e. Because of the structure of $\tilde{E}_0$ and $\tilde{E}_1$ we have

$$\begin{aligned} \operatorname{Ker} \tilde{E}_0 &= \operatorname{span}\{e_1, ..., e_{n_e - \gamma_0}\}, & (7) \\ \operatorname{Im} \tilde{E}_1 &= \operatorname{span}\{e_1, ..., e_{\gamma_1}\}. & (8) \end{aligned}$$

The first homology group H_1 is spanned by the classes

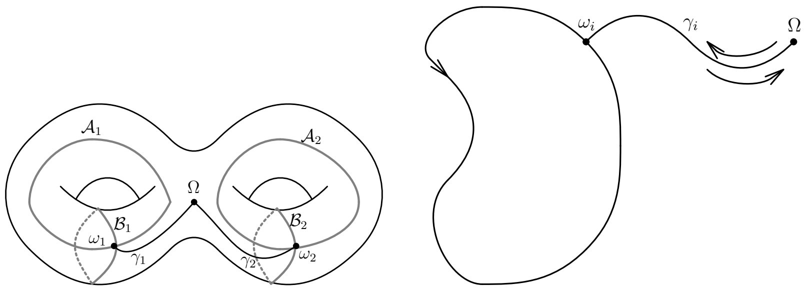
$$[e_{\gamma_1+1}],\ [e_{\gamma_1+2}],\ ...,\ [e_{n_e-\gamma_0}] \tag{9}$$

in which $[f]$ is the class of a representant f. In fact, we could replace the representant f by another one

$$\tilde{f} = f + \sum_{i=1}^{\gamma_1} \lambda_i e_i\ . \tag{10}$$

After that process, the classes from relation (9) provides us with a set of pairs of closed curves

$$\tilde{\mathcal{A}}_1, \tilde{\mathcal{B}}_1, ..., \tilde{\mathcal{A}}_g, \tilde{\mathcal{B}}_g \tag{11}$$



(a) (b)

Figure 5: (a)Connect the homology bases with Ω by means of approach paths (b)Follow the approach path γ_i then the closed curve and finally traverse γ_i backward.

such that each pair $(\tilde{\mathcal{A}}_i, \tilde{\mathcal{B}}_i)$ intersect only at a single point ω_i as illustrated in Fig. 5(a). In order to have the preliminary canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$,..., $\mathcal{A}_g$, $\mathcal{B}_g$ from the curves $\tilde{\mathcal{A}}_1$, $\tilde{\mathcal{B}}_1$,$\cdots$, $\tilde{\mathcal{A}}_g$, $\tilde{\mathcal{B}}_g$, we use the following algorithm

Algorithm: Using approach paths

step 1 : Refine the triangular mesh $\mathcal{M}$ so that the edges of the curves $(\tilde{\mathcal{A}}_i, \tilde{\mathcal{B}}_i)$ become edges of the mesh.

step 2 : Choose as basepoint any point Ω of the surface which does not reside on those curves.

step 3 : Use Dijkstra algorithm in order to find a path γ_i joining the basepoint Ω and the crossing point ω_i as illustrated in Fig. 5(a).

step 4 : At this point we have a set of loops $\mathcal{L}_i$. Each loop $\mathcal{L}_i$ is obtained by first traversing the approach path γ_i, then the closed curve $\mathcal{C}_i := \tilde{\mathcal{A}}_i$ or $\mathcal{C}_i := \tilde{\mathcal{B}}_i$ and finally traversing γ_i backward as depicted in Fig. 5(b).

step 5 : Shift the loops $\mathcal{L}_i$ homotopically so that they traverse triangles internally as in Fig. 6(b). Use Reidemeister moves (see Appendix) so that the loops have only intersections at the basepoint Ω.

3.2 Numerical realization

For the reduction of an integer matrix A into normalized canonical form as in relation (5), we distinguish three row operations:
(Op1) Swap the i-th row R_i and the j-th row R_j of A.
(Op2) Multiply the i-th row R_i by (-1).
(Op3) Replace the i-th row R_i by $R_i + qR_k$ where q is a given integer.

Note that all those three operations are invertible. In fact, the first two are self-invertible while the inverse of the last one is $R_i - qR_k$.
Similar elementary operations can be done for the columns of A. The reduction into normalized form is processed in two steps. First, we reduce A into Smith normal form

$$\tilde{A} = GAF \tag{12}$$

and then we perform some column swappings. Let us describe now how to reduce a matrix A having a form like in (2) into a matrix of the form like in (5). Suppose the matrix composed of the first c rows and the first c columns has already been reduced. The next step are the following

1. Apply the above elementary operations so that $\alpha := A_{cc}$ divides the remaining entries in the c-th row and the c-th column. That is

$$\exists k_i, h_j \in \mathbf{Z} \quad \text{with} \quad A_{ic} = k_i\alpha \quad \text{and} \quad A_{cj} = h_j\alpha \quad \forall\, i, j > c\,. \tag{13}$$

2. Make all those entries zero by applying the third operation (Op3). That is, for all $i, j > c$, replace row R_i by $R_i - pR_c$ with $p = A_{ic}/\alpha$ and column C_j by $C_j - qC_c$ with $q = A_{cj}/\alpha$.

Of course, one has to perform column swapping operations to have the final form in (5). Since we are interested in the matrices of change of bases G^{-1} and F in (12), we should store the parameters for the elementary operations so that we do not need to explicitly compute the inverse of the integer matrices at the end of the process.
The difficulty of directly applying the above process to the incidence matrices E_0 and E_1 is that we have to do them simultaneously. Observe that we use the same matrix M_e of change of bases in equation (6) for both E_0 and E_1. For that end, we execute the following steps.

1. Reduce E_0 in normalized canonical form:

$$\overline{E}_0 = M_v^{-1} E_0 \overline{M}_e. \tag{14}$$

2. Apply the change of bases $\overline{M}_e$ to E_1:

$$\overline{E}_1 = \overline{M}_e^{-1} E_1. \tag{15}$$

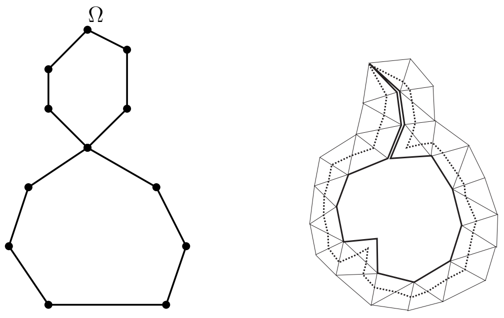
3. Reduce $\overline{E}_1$ in normalized canonical form:

$$\tilde{E}_1 = \tilde{M}_e^{-1} \overline{E}_1 M_t = (\overline{M}_e \hat{M}_e)^{-1} E_1 M_t. \tag{16}$$

4. Define

$$\tilde{E}_0 := \overline{E}_0 \tilde{M}_e = M_v^{-1} E_0 (\overline{M}_e \tilde{M}_e). \tag{17}$$

By introducing $M_e := \overline{M}_e \tilde{M}_e$, we obtain the relation (6).



(a) (b)

Figure 6: (a)Impossible double nodes inside one individual loop (b)Initial loop $\mathcal{L}_e$ in bold line and homotopically shifted loop $\mathcal{L}'_e$ in dotted line.

3.3 Bases of the fundamental group

Now, we would like to describe a second method of finding the canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$,...,$\mathcal{A}_g$, $\mathcal{B}_g$ by using the fundamental group. In order to find the basis of the fundamental group $\pi_1(\mathcal{M})$ [10] of a two dimensional simplicial complex $\mathcal{M}$, we follow the next major steps.

Algorithm: Bases of $\pi_1(\mathcal{M})$

step 1	:	Choose an arbitrary point Ω of $\mathcal{M}$ as a basepoint.
step 2	:	Find a list of loops (a sequence of edges starting and terminating at the basepoint Ω) $\mathcal{L}_e$, $e \in S$.
step 3	:	Shift the loops homotopically so that they traverse triangles as in Fig. 6(b).
step 4	:	Use Reidemeister moves so that the loops have only intersections at the basepoint Ω.

Let us describe the way of achieving step 2 by following the standard approach [10, 16, 19] requiring the use of a spanning tree. Let G be the edge-vertex graph which is generated from the mesh $\mathcal{M}$. We use the Breadth First Search (BFS) algorithm [2] to find a spanning tree T [12] of G which is rooted at the basepoint Ω. Now we would like to describe how the set of edges S is found. We introduce a subgraph Γ of G. As an initialization we define S to be the empty set and Γ to be the spanning tree T. We perform the following updating process of Γ and S.

Search for an edge $e = [v, w]$ from the set $G \setminus \Gamma$ for which there exists a triangle $[p, v, w]$ of the mesh $\mathcal{M}$ such that both of the edges $[p, v]$ and $[p, w]$ belong to the subgraph Γ. We distinguish now two cases depending on the success of search. If such an edge exists, we include it in the subgraph Γ:

$$\Gamma := \Gamma \cup [v, w]. \tag{18}$$

Otherwise we pick any edge $e = [v, w]$ from $G \setminus \Gamma$ and then we include it in the list S. We

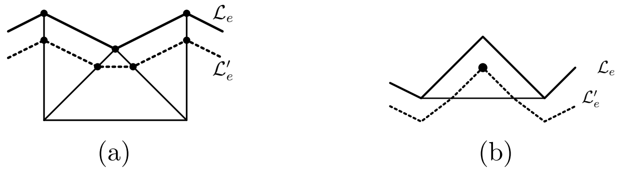


Figure 7: (a)First type of loop shifting (b)Second type of loop shifting

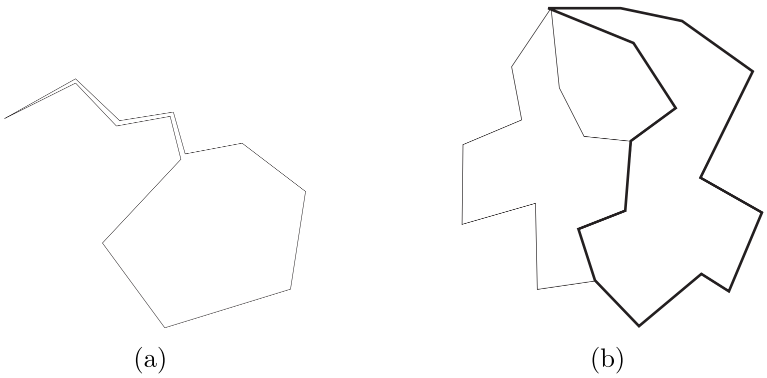


Figure 8: (a)A loop with double edges (b)Two loops with common edges.

terminate the updating of S and Γ when the graph Γ becomes the same as the graph G. Afterward, for each edge $e = [v, w]$ of S we generate a loop $\mathcal{L}_e$ in the following way. Find a path p_v (resp. p_w) which belongs to the spanning tree T and which starts at v (resp. w) and which terminates at the basepoint Ω. We define the loop $\mathcal{L}_e$ corresponding to the edge e as

$$\mathcal{L}_e := p_v e p_w^{-1} \tag{19}$$

which is p_v followed by e and then by the inverse p_w. The loops $\mathcal{L}_e$ are also commonly termed '*generators*' of the fundamental group $\pi_1(\mathcal{M})$.

3.4 Single curve intersection

Before describing the content of step 3, we would like to mention a few features of the loops $\mathcal{L}_e$ which result from step 2. There are mainly two common problems after the execution of step 2. The first one is a local one which happens for an individual generator without considering the other generators. The second problem is a global one in which two different loops $\mathcal{L}_e$ and $\mathcal{L}_f$ could interfere. They have intersections away from the basepoint. More accurately, there could exist

- Double edges: different edges which have the same metric information as in Fig. 8(a) or
- Double nodes which are not at the basepoint (Fig. 8(b)).

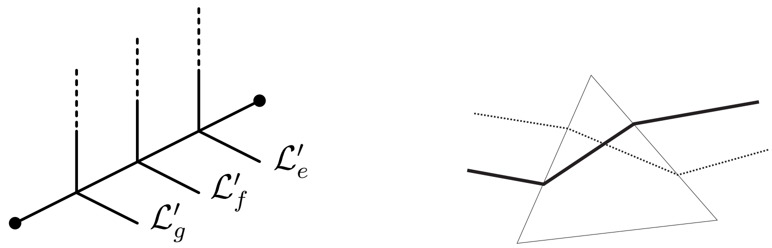


Figure 9: (a)Uniformly splitting an edge traversed by loops (b)Loop intersection inside a triangle.

Now we would like to describe step 3 which consists in homotopically transforming the loops $\mathcal{L}_e$ so that those problems do not occur. For a given loop $\mathcal{L}_e$, our objective is to generate another loop $\mathcal{L}'_e$ which starts and terminates at the basepoint Ω and which traverses the triangles on the left of the original loop $\mathcal{L}_e$ as in Fig. 6(b). The surface being orientable, it is not difficult to decide whether an emanating edge from the loop $\mathcal{L}_e$ is on its left or on its right by simple "combinatorial" strategy (ie. without any metric information). That can be done by orienting all three local vertices of a triangle in anti-clockwise direction. Now let us denote by $\mathcal{R}$ the list of all edges which emanate from a node $\theta \neq \Omega$ of the loop $\mathcal{L}_e$ and which go to the left hand side.
For that end, we need to generate new nodes which we categorize in two types. The first type of new nodes X resides on an edge $e = [\theta, B]$ of $\mathcal{R}$ and its distance from the node loop θ is controlled by a prescribed positive parameter μ as $X = \mu B + (1 - \mu)\theta$. As an illustration, we find in Fig. 7(a) a dotted curve $\mathcal{L}'_e$ obtained by shifting a solid curve $\mathcal{L}_e$ using the nodes of first type.
The second type of node is generated when two consecutive edges of a triangle belong to the loop $\mathcal{L}_e$. In such a case a node inside a triangle is generated as in Fig. 7(b). Its coordinates can also be controlled by a parameter μ in which you use barycentric coordinates. The generation of the loop $\mathcal{L}'_e$ consists therefore in joining those newly generated nodes.
We note the obvious fact that this process generates a loop which is homotopically equivalent to the first one. We would like to mention that this process solves the first problem of coinciding edges. That is due to the fact that a loop such as the one depicted in Fig. 6(a) can never occur after execution of step 2. Let us recall from (19) that the edges of the loop $\mathcal{L}_e$ from step 2 follow the shortest paths on the spanning tree T. Therefore, once two nodes of $\mathcal{L}_e$ coincide, the following paths will also coincide till the root Ω of the spanning tree. We would like to notice that the above homotopic shifts do not solve the global problem of interference of several different loops.
Let us now propose a remedy for the global problem. The first stage of the remedy consists in uniformization in which we search for the list of edges which are traversed by loops. We shift afterwards the nodes on the edges in such a way that they are equispaced as in Fig. 9(a). After this stage, no two different nodes (other than the basepoint) could have the same coordinates (ie. there is no double node). But there could still exist the problem of

intersection of two different loops which could only occur inside triangles as in Fig. 9(b). After the execution of step 3, we have the case of "regular system" of curves which are described in [6]. We can apply a series of Reidemeister moves [14, 1] to obtain our final generators and that is exactly the purpose of step 4.

4 Improvements of the canonical curves

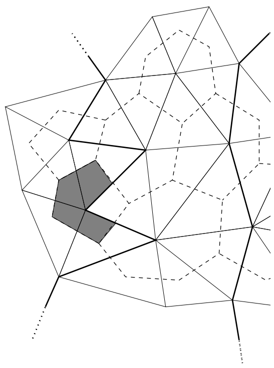
After a direct application of the formerly described algorithms, the resulting canonical curves could be too long or unnecessarily complicated. That could lead to undesirable splitting of the topological surface $\mathcal{M}$ as in Fig. 10(b). In order that we can apply the subsequent algorithm efficiently, we need to shorten the lengths of the canonical curves. Suppose the canonical curves obtained from the former sections are $\overline{\mathcal{A}}_1$,$\overline{\mathcal{B}}_1$,...,$\overline{\mathcal{A}}_g$, $\overline{\mathcal{B}}_g$. In this section, we would like to describe an algorithm to improve them and to obtain the final canonical curves $\mathcal{A}_1$, $\mathcal{B}_1$,..., $\mathcal{A}_g$, $\mathcal{B}_g$. Before describing the improvement algorithm, let us introduce the notion of weighted split graph which can be obtained from the edge-vertex graph and the current canonical curves.

4.1 Weighted split graph

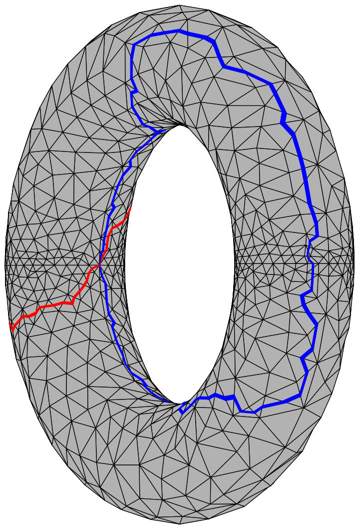
Suppose we have a few curves $\mathcal{S}_1, \mathcal{S}_2..., \mathcal{S}_{2g}$ on the triangulated surface $\mathcal{M}$. Every curve $\mathcal{S}_i$ is composed of a sequence of consecutive edges of the mesh $\mathcal{M}$. For every index i from $\{1, ..., 2g\}$, let us describe [5] how a weighted graph H_i is defined by specifying its nodes, edges and weights. Consider the dual graph H of the edge-vertex graph G. At this first stage, the surface $\mathcal{M}$ is split into different faces F_k which are delineated by the edges of H. By inserting the edges of $\mathcal{S}_k$ for all k different from i inside the relevant faces F_k, we obtain a splitting of the topological surface $\mathcal{M}$ into several connected components C_1, C_2,..., C_N. In Fig. 10(a) , we see an illustration of the graph H_i where thin segments represent the edge-vertex graph G while the dual graph H is depicted with dotted lines and the canonical curves are shown in bold faces. The highlighted region is an example of a connected component. The nodes of the weighted split graph H_i are then defined to be the connected components.

With the nodes of H_i in place, let us now define its edges. Consider two graph-nodes N_p and N_q whose corresponding connected components are C_p and C_q respectively. Consider the parent faces F_p and F_q of the connected components C_p and C_q. Let us denote by $\mathcal{T}_p$ (resp. $\mathcal{T}_q$) the set of triangles of F_p (resp. F_q) which are in the connected component C_p (resp. C_q). Between the two graph-nodes N_p and N_q of the graph H_i, we introduce a graph-edge e_{pq} of H_i if the sets $\mathcal{T}_p$ and $\mathcal{T}_q$ share a common triangle t_{pq}.

Now let us define the weight w_{pq} that is assigned to the graph-edge e_{pq}. Two nodes Ω_p and Ω_q of the triangle t_{pq} must be the centers of the parent faces F_p and F_q. The weight w_{pq} assigned to the edges e_{pq} will then be the Euclidean distance between Ω_p and Ω_q.



(a)



(b)

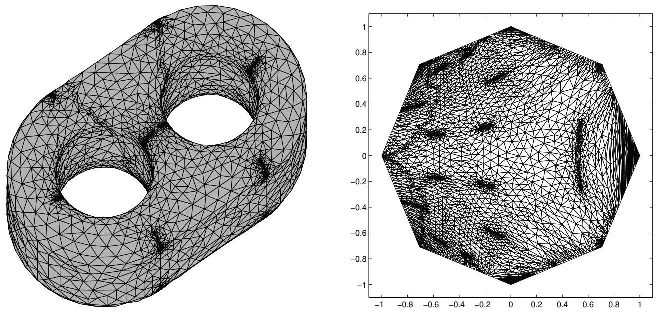
Figure 10: (a) Dual graph split by the canonical curves (b) Undesirable canonical curves.

4.2 Improvement algorithm

The improvement of the canonical curve is processed recursively as follow. Suppose the current canonical curves are $\mathcal{S}_1$, $\mathcal{S}_2$, ...,$\mathcal{S}_{2g}$ with g being the genus of the surface. For a given index $i = 1, ..., 2g$, we would now like to describe hot to find the improved canonical curve $\mathcal{S}'_i$. Using the weighted graph H_i, we can search for the shortest path from the starting and the terminating connected components of S_i. The new path $\mathcal{S}'_i$ will therefore consist of the thus defined path.
Notice that after the search for the shortest path in the weighted graph H_i, you only obtain a sequence of connected components. Care must therefore be taken in order to convert that sequence to a list of nodes because the path S_i should not touch the other canonical curves except at the basepoint Ω.

5 Global parameterization

Suppose that we have an orientable surface $\mathcal{M}$ without boundary and a set of canonical curves. We would like to have a single parameterization of the whole surface $\mathcal{M}$ from a planar parameter domain as illustrated in Fig. 11.
Thus, we would like to determine a planar parameter domain and the corresponding parameterization. For the parametrization, we follow the idea of Floater [8] which solves a linear system to find coordinates of the nodes of the 2D-mesh. We would like now to point out how it applies to global parameterization. Let us recall that the canonical curves of the surface $\mathcal{M}$ have a single common point Ω. Consequently, if we parameterize the polygonal



(a) (b)

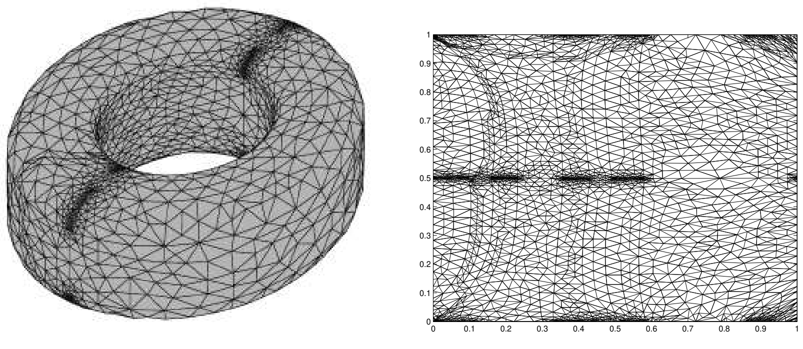
Figure 11: (a)Surface with genus 2 (b)Parameterization on $aba^{-1}b^{-1}cdc^{-1}d^{-1}$

disk $\mathbf{P}$ to $\mathcal{M}$ then all the 2D-corner vertices w_1, w_2, ..., w_{2g-1} of $\mathbf{P}$ map to the same vertex Ω of the surface mesh $\mathcal{M}$. More precisely, the basepoint Ω is mapped $2g$ times if we deal with genus g. That means we have $2g$ different indices but those $2g$ points all have the same coordinates. Similarly, the points $\overline{w_1}$ and $\overline{w_2}$ which are portrayed in Fig. 4(b) maps to the same 3D points A of the surface mesh $\mathcal{M}$. For that reason, the point A has to be repeated twice. The vertices of the polygonal disk are chosen as

$$w_s = [\cos(s\pi/2g), \sin(s\pi/2g)] \quad \text{forall} s = 0, 1, ...2g-1\,. \tag{20}$$

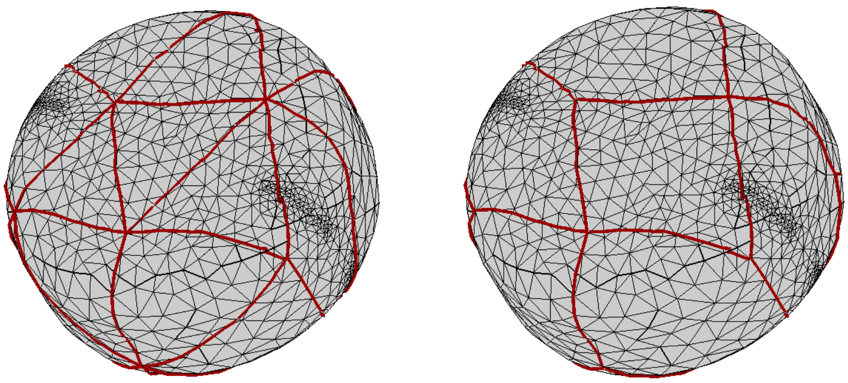
6 Constrained quadrangulation

In this section, we will describe a way to decompose a surface $\mathcal{M}$ into pieces of four-sided domains. In order to facilitate the presentation, we suppose that we have a parametrization $\mathcal{P}$ in disposition and that $\mathcal{M}$ is of genus zero and thus the parameter domain is a rectangle.



(a) (b)

Figure 12: (a)Surface with genus 1 (b)Parameterization on $aba^{-1}b^{-1}$



(a) (b)

Figure 13: (a) Triangular decomposition (b) Quadrilateral decomposition

6.1 Globally smooth surfaces

Suppose for now that the surface $\mathcal{M}$ to be approximated is globally smooth in the sense that it does not have very sharp edges. Our way of segmenting $\mathcal{M}$ into several four-sided pieces is done into two steps as explained in Fig. 13. The first one consists in splitting $\mathcal{M}$ into large curved triangles. Afterwards, we convert the resulting triangular decomposition $\mathcal{T}$ into four-sided decomposition $\mathcal{Q}$. For one thing, the Delaunay approaches are better understood in the context of triangulations than quadrangulations and it is very simple to split a triangle without introducing any hanging node and without increasing the number of new elements significantly. For another, we have already a parameterization $\mathcal{P}$ in disposition and we can thus apply the idea of Riemannian Delaunay triangulation as we did in [18].
In order to convert a triangulation $\mathcal{T}$ to a quadrangulation $\mathcal{Q}$, we follow the algorithm in [17] or we find a Hamiltonian path on the edge graph of the large triangular mesh and we merge two consecutive triangles on the path. Let us recall that a Hamiltonian path [2] from a graph G is a path which traverse all the nodes and each node is traversed only once. Now we would like to describe our decomposition algorithm to obtain the large triangulation $\mathcal{T}$. Since the method of generalized Delaunay triangulation was already presented in full detail in many literatures, we need only to specify the criteria for splitting an edge (Fig. 14(a)) and for inserting a node inside a triangle (Fig. 14(b)).

Definition 1 For a curve $\mathcal{C}$ defined on an interval $[a, b]$, we define its linear curve distortion $\epsilon(\mathcal{C})$ as

$$\epsilon(\mathcal{C}) := \frac{1}{\|AB\|} \sum_{i\in\mathcal{I}} \|\mathcal{C}(t_i) - \mathcal{L}(t_i)\|, \text{where} \quad A := \mathcal{C}(a), \quad B := \mathcal{C}(b) \quad \text{and} \tag{21}$$

$$\mathcal{L}(t_i) := \lambda_i A + (1-\lambda_i)B \quad \text{with} \quad \lambda = (t_i - a)/(b-a) \quad \text{and} \tag{22}$$

t_i are some values taken within $[a, b]$.

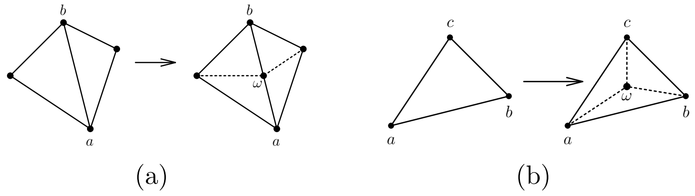


Figure 14: Insertion of a new node: (a) edge cutting (b) triangle subdivision

This definition can be extended to the case of linear surface distortion $\mu(S)$ of a surface S defined on a triangle $[a, b, c]$ where we need to replace $\mathcal{L}(t_i)$ by

$$\mathcal{L}(t_i) := \lambda_i^a A + \lambda_i^b B + \lambda_i^c C \quad \text{with} \tag{23}$$

λ_i^a, λ_i^b, λ_i^c being the barycentric coordinates.

The determination of the large triangulation $\mathcal{T}$ is done recursively by starting from an initial coarse triangulation $\mathcal{T}^{(0)}$. The large triangulation $\mathcal{T}^{(k+1)}$ is obtained by $\mathcal{T}^{(k)}$ by generalized Delaunay point insertion: we split an edge (resp. triangle) of $\mathcal{T}^{(k)}$ if its linear curve (resp. surface) distortion exceeds some prescribed accuracy ϵ_0 (resp. μ_0).
The initial coarse large triangulation $\mathcal{T}^{(0)}$ can be done manually by picking a few vertices on the surface $\mathcal{M}$. We recall that the quadrangulation conversion is only feasible if the number of triangles of $\mathcal{T}$ is even. If we start from a coarse triangulation $\mathcal{T}^{(0)}$ with an even number of triangles, then applying the mesh operations in Fig. 14 will keep the parity of the number of triangles even. In order to have the parametric representations, we use the method in section 5. For genus zero surfaces, we need two parameterizations $\mathcal{P}_1$ and $\mathcal{P}_2$. The first parameterization $\mathcal{P}_1$ consists of a split surface and parameterization $\mathcal{P}_2$ is a representation of the surface in the vicinity of the split curve $(\mathcal{A}, \mathcal{B})$ as in Fig. 7.2. For surfaces of higher genera $g \geq 1$, we proceed similarly but the parameterizations are now neighbors of the canonical curves and the basepoint.

6.2 Dealing with sharp edges

In the former discussion, we have described the quadrangulation process for a surface $\mathcal{M}$ which is supposed to be globally smooth. For surfaces presenting corners or sharp edges, the problem is more complicated because we do not want that a sharp edge traverses a four-sided surface patch within its internal domain. Otherwise, the resulting four-sided patch in the surface M could have internal irregularities and therefore no possible local smooth approximation is conceivable. For that end, we use similar ideas as CDT (Constrained Delaunay Triangulation). Let us recall that a CDT [4, 13] is a method for generating a triangulation which interpolates a prescribed set of points and line segments. That is, those points and line segments will be the vertices and the edges of the final triangulation $\mathcal{T}$. We want to carry over that idea to the case of large triangular subdivision. There are several numerical methods which describe an approach for detecting sharp edges whose corresponding parameter values can be determined simultaneously. Therefore, we put constraints on the large triangular subdivisions $\mathcal{T}$ in order that the curves represented by the sharp edges become edges of the final large triangulation $\mathcal{T}$.

7 Surface fitting with C^0 joint

Let us suppose we have K patches. We would like to approximate/represent each four-sided surface S_r by a Bézier surface. For each $r = 1, ..., K$ we have therefore the representation of S_r:

$$\mathbf{X}_r(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{n} \mathbf{b}_{ij}^r B_i^n(u) B_j^n(v) \,. \tag{24}$$

From now on, we want to fit a Bézier surface $\mathbf{b}_{ij}$ for given samples $(u, v)_k$ $\mathbf{P}_k \in M$. We will drop the index r when no confusion is possible.
After some lexicographical reordering, we could have the following representation of the relation (24).

$$\mathbf{X}(u, v) = \sum_{s=0}^{N} \mathbf{d}_s R_s(u, v). \tag{25}$$

Let us denote by $\mathbf{D}$ the unknown sequence $\mathbf{d}_s$:

$$\mathbf{D} = [\mathbf{d}_0, ..., \mathbf{d}_N] \,. \tag{26}$$

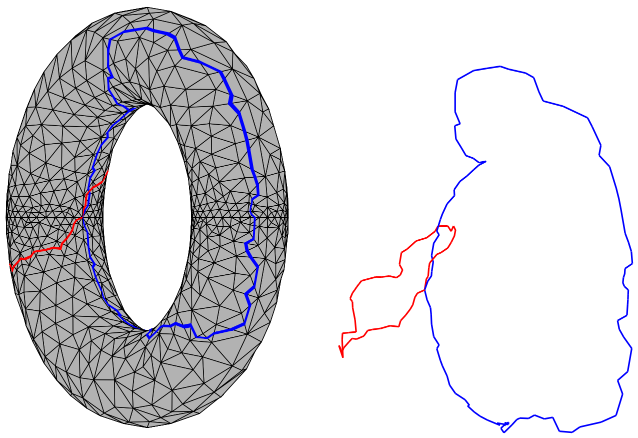
In order that we have global C^0, the Bézier patches have to abut at the interfaces. Therefore we have to add the condition that the control points at the boundaries $b_{0\star}$, $b_{n\star}$, $b_{\star 0}$, $b_{\star n}$ are prescribed. These control points are then interpolated by two adjacent Bézier surfaces. We are going to describe later on how to determine their values but for now we assume that they are known explicitly. By excluding the external control points from $\mathbf{d}_i$, we have the remaining unknowns. Let us denote their sequence by $\mathbf{E}$ and their number by $R := (N+1)^2 - 4N$. The problem is then locally reduced to a least-square approximation:

$$\sum_{r=0}^{R} \|\mathbf{P}_r - \mathbf{X}_E(u_r, v_r)\| \to \min_{\mathbf{E}} \tag{27}$$

8 Numerical results

In this section, we would like to describe some results of the former methods when applied to some meshes. First, we would like to show the developments of the canonical curves in three stages:

- Before applying the Reidemeister moves,
- After application of the Reidemeister moves which give us the preliminary canonical curves,



(a) (b)

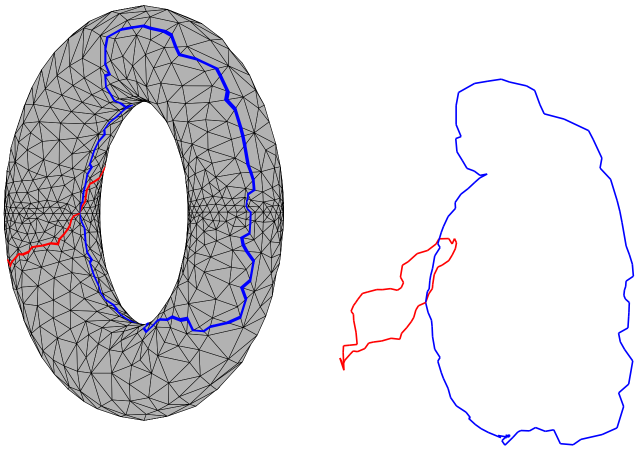
Figure 15: Before Reidemeister moves

- The ultimate canonical curves after optimization.

We will describe our results with the help of two surfaces which have respectively genus 1 and 2 (torus and pretzel). In the figures, we display on the left hand sides the canonical curves which are traced on the underlying surface meshes. We find on the right hand sides the canonical curves without surfaces in order to facilitate the visualization of the curve properties. In Fig. 15, we can see that the canonical curves still have an intersection which is not at the basepoint. After application of the Reidemeister moves, there remains only one intersection at the basepoint but the curves are still very undesirable as seen in Fig. 16. By applying the optimization algorithm that we described in section 4, we obtain the ultimate canonical curves (Fig. 17) in which we have one single intersection and the quality of the curves has been optimized homotopically.
The same process has been successfully applied to a surface of genus two (see Fig(s) 18, 19, 21). We can see a magnification of the results after Reidemeister moves in the neighborhood of the basepoint in Fig. 20. As Fig. 20(a) clearly shows , there are still several curve crossings which can be eliminated with the help of Reidemeister moves. In Fig. 20(b) we see only one position where we have curve crossings which are exactly at the basepoint.
The second result consists in reconstructing surfaces from three surface meshes which have respectively 8288, 11656 and 21872 triangles. We can see the surface meshes together with the splittings in Figs 22(a), 23(a), 24(a). The corresponding surface patches are located on the right hand side.

References

[1] C. Adams, The knot book, Freeman and Co., New York, 1994.



(a) (b)

Figure 16: After Reidemeister moves

[2] S. Althoen, R. Bumcrot, Introduction to discrete mathematics, PWS-KENT publishing company, Boston, 1988.

[3] E. Bloch, A first course in geometric topology and differential geometry, Birkhäuser, Boston, 1997.

[4] L. P. Chew, Constrained Delaunay triangulations, Algorithmica **4**, No. 1, 97–108, 1989.

[5] E. Colin de Verdière, Raccourcissement de courbes et décomposition de surfaces, Ph.D. thesis, Université Paris 7, 2003.

[6] M. De Graaf, A. Schrijver, Making curves minimally crossing by Reidemeister moves, J. Comb. Theory, Ser. B **70**, No.1, 134–156, 1997.

[7] R. Engelking, K. Sieklucki, Topology: a geometric approach **4**, Heldermann Verlag, Berlin, 1992.

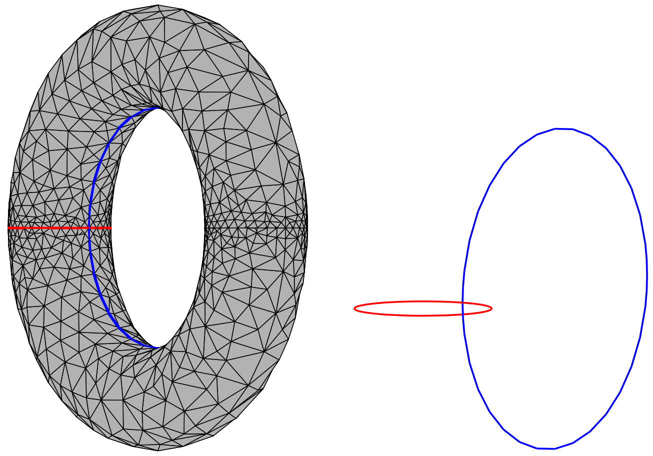
[8] M. Floater, K. Hormann, Parameterization of triangulations and unorganized points, Iske, Armin (ed.) et al., in: Tutorials on multiresolution in geometric modelling, European summer school lecture notes, Munich Univ. of Technology, Germany, 2001.

[9] A. Fomenko, Visual geometry and topology, Springer-Verlag, Berlin, 1993.

[10] A. Fomenko, S. Matveev, Algorithmic and computer methods for three-manifolds, Kluwer Academic Publishers, Dordrecht, 1997.

[11] A. Fomenko, T. Kunii, Topological modeling for visualization, Springer, Tokyo, 1997.

[12] D. Jungnickel, Graphs, networks and algorithms, Springer-Verlag, Berlin, 1999.



(a) (b)

Figure 17: After combinatorial optimization

[13] M. Kallmann, H. Bieri, D. Thalmann, Fully dynamic constrained Delaunay triangulations, in: Geometric modelling for scientific visualization, G. Brunnett, B. Hamann, H. Mueller, L. Linsen (Eds.), Springer-Verlag, Heidelberg, Germany, 241–257, 2003.

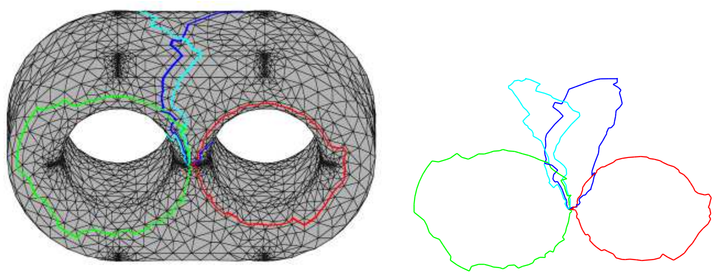
[14] C. Livingston, Knot theory, The Carus mathematical monographs, The mathematical association of America, Washington, 1993.

[15] W. Massey, A basic course in algebraic topology, Springer Verlag, New-York, 1991.

[16] S. Rees, L. Soicher, An algorithtmic approach to fundameental groups and covers of combinatorial cell complexes, J. Symb. Comput. **29**, No.1, 59–77, 2000.

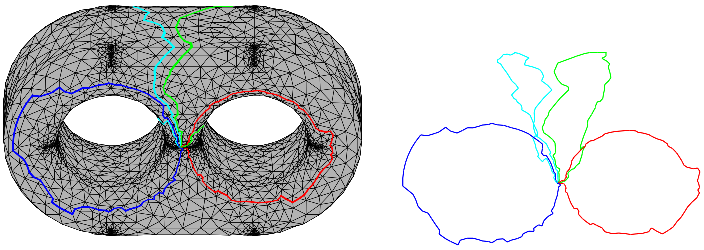
[17] S. Ramaswami, P. Ramos, G. Toussaint, Converting triangulations to quadrangulations, Comput. Geom. **9**, No. 4, 257–276, 1998.

[18] M. Randrianarivony, G. Brunnett, Mesh generation on piecewise Riemannian surfaces, to appear.



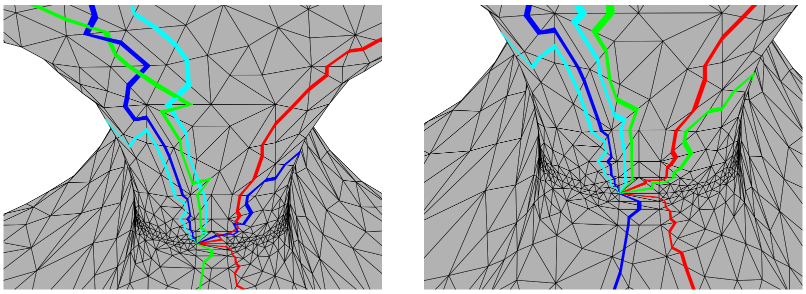
(a) (b)

Figure 18: Before Reidemeister moves



(a) (b)

Figure 19: After Reidemeister moves



(a) (b)

Figure 20: Removing curve-crossings with the help of Reidemeister moves

[19] J. Stillwell, Classical topology and combinatorial group theory, Second edition, Springer Verlag, New York, 1993.

[20] R. Stöcker, H. Zieschang, Algebraische Topologie, B. G. Teubner, Stuttgart, 1988.

[21] M. Vanco, A direct approach for segmentation of unorganized points and recognition of simple algebraic surfaces, Ph. D. thesis, Technische Universität Chemnitz, 2003.

[22] T. Varady, P. Benko, G. Kos, Reverse engineering regular objects: simple segmentation and surface fitting procedures, Int. J. Shape Model. **4** , 127–141, 1998.

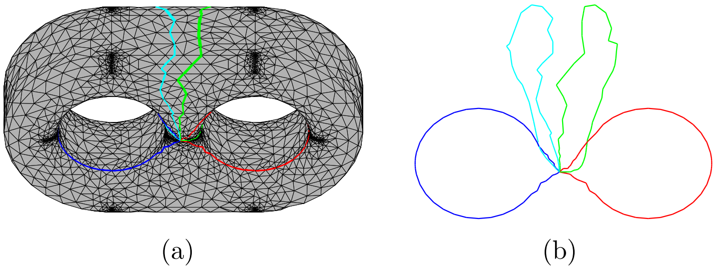


Figure 21: After combinatorial optimization

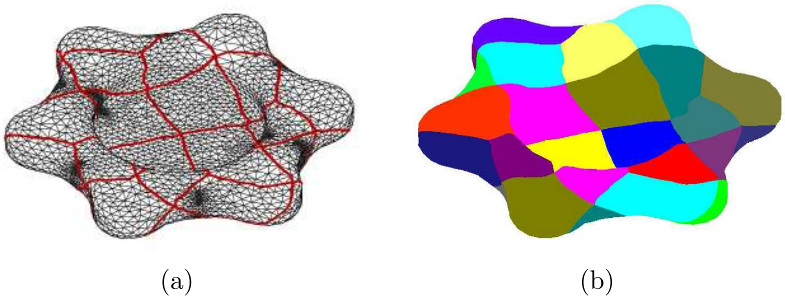


Figure 22: (a)Mesh with 8288 triangles (b)Approximation with 40 patches

Other titles in the SFB393 series:

03-01 E. Creusé, G. Kunert, S. Nicaise. A posteriory error estimation for the Stokes problem: Anisotropic and isotropic discretizations. January 2003.

03-02 S. I. Solov'ëv. Existence of the guided modes of an optical fiber. January 2003.

03-03 S. Beuchler. Wavelet preconditioners for the p-version of the FEM. February 2003.

03-04 S. Beuchler. Fast solvers for degenerated problems. February 2003.

03-05 A. Meyer. Stable calculation of the Jacobians for curved triangles. February 2003.

03-06 S. I. Solov'ëv. Eigenvibrations of a plate with elastically attached load. February 2003.

03-07 H. Harbrecht, R. Schneider. Wavelet based fast solution of boundary integral equations. February 2003.

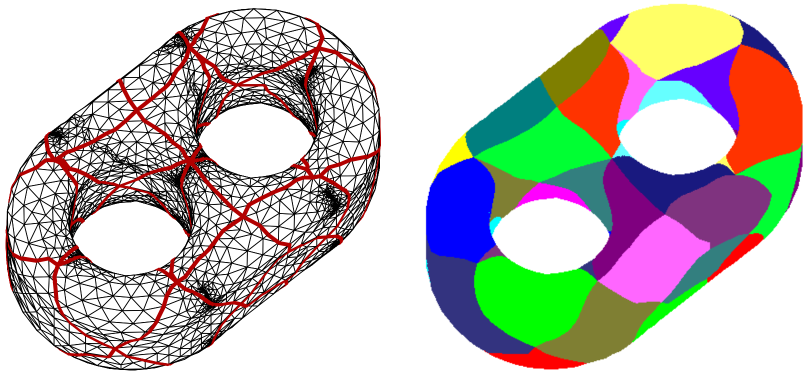
03-08 S. I. Solov'ëv. Preconditioned iterative methods for monotone nonlinear eigenvalue problems. March 2003.

03-09 Th. Apel, N. Düvelmeyer. Transformation of hexahedral finite element meshes into tetrahedral meshes according to quality criteria. May 2003.

03-10 H. Harbrecht, R. Schneider. Biorthogonal wavelet bases for the boundary element method. April 2003.

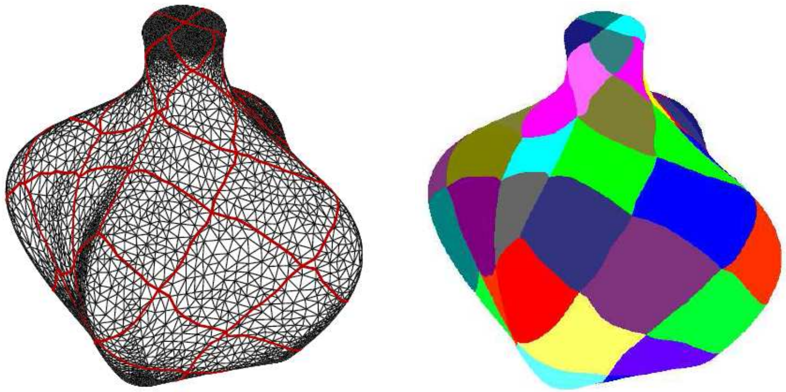
03-11 T. Zhanlav. Some choices of moments of refinable function and applications. June 2003.

03-12 S. Beuchler. A Dirichlet-Dirichlet DD-pre-conditioner for p-FEM. June 2003.



(a) (b)

Figure 23: (a)Mesh with 11656 triangles (b)Approximation with 32 patches



(a) (b)

Figure 24: (a)Mesh with 21872 triangles (b)Approximation with 52 patches

03-13 Th. Apel, C. Pester. Clément-type interpolation on spherical domains - interpolation error estimates and application to a posteriori error estimation. July 2003.

03-14 S. Beuchler. Multi-level solver for degenerated problems with applications to p-version of the fem. *(Dissertation)* July 2003.

03-15 Th. Apel, S. Nicaise. The inf-sup condition for the Bernardi-Fortin-Raugel element on anisotropic meshes. September 2003.

03-16 G. Kunert, Z. Mghazli, S. Nicaise. A posteriori error estimation for a finite volume discretization on anisotropic meshes. September 2003.

03-17 B. Heinrich, K. Pönitz. Nitsche type mortaring for singularly perturbed reaction-diffusion problems. October 2003.

03-18 S. I. Solov'ëv. Vibrations of plates with masses. November 2003.

03-19 S. I. Solov'ëv. Preconditioned iterative methods for a class of nonlinear eigenvalue problems. November 2003.

03-20 M. Randrianarivony, G. Brunnett, R. Schneider. Tessellation and parametrization of trimmed surfaces. December 2003.

04-01 A. Meyer, F. Rabold, M. Scherzer. Efficient Finite Element Simulation of Crack Propagation. February 2004.

04-02 S. Grosman. The robustness of the hierarchical a posteriori error estimator for reaction-diffusion equation on anisotropic meshes. March 2004.

04-03 A. Bucher, A. Meyer, U.-J. Görke, R. Kreißig. Entwicklung von adaptiven Algorithmen für nichtlineare FEM. April 2004.

04-04 A. Meyer, R. Unger. Projection methods for contact problems in elasticity. April 2004.

04-05 T. Eibner, J. M. Melenk. A local error analysis of the boundary concentrated FEM. May 2004.

04-06 H. Harbrecht, U. Kähler, R. Schneider. Wavelet Galerkin BEM on unstructured meshes. May 2004.

04-07 M. Randrianarivony, G. Brunnett. Necessary and sufficient conditions for the regularity of a planar Coons map. May 2004.

04-08 P. Benner, E. S. Quintana-Ortí, G. Quintana-Ortí. Solving Linear Matrix Equations via Rational Iterative Schemes. October 2004.

04-09 C. Pester. Hamiltonian eigenvalue symmetry for quadratic operator eigenvalue problems. October 2004.

04-10 T. Eibner, J. M. Melenk. An adaptive strategy for hp-FEM based on testing for analyticity. November 2004.

04-11 B. Heinrich, B. Jung. The Fourier-finite-element method with Nitsche-mortaring. November 2004.

04-12 A. Meyer, C. Pester. The Laplace and the linear elasticity problems near polyhedral corners and associated eigenvalue problems. December 2004.

04-13 M. Jung, T. D. Todorov. On the Convergence Factor in Multilevel Methods for Solving 3D Elasticity Problems. December 2004.

05-01 C. Pester. A residual a posteriori error estimator for the eigenvalue problem for the Laplace-Beltrami operator. January 2005.

05-02 J. Badía, P. Benner, R. Mayo, E. Quintana-Ortí, G. Quintana-Ortí, J. Saak. Parallel Order Reduction via Balanced Truncation for Optimal Cooling of Steel Profiles. February 2005.

05-03 C. Pester. CoCoS – Computation of Corner Singularities. April 2005.

05-04 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Einige Elemente, Fehlerschätzer und Ergebnisse. April 2005.

05-05 P. Benner, J. Saak. Linear-Quadratic Regulator Design for Optimal Cooling of Steel Profiles. April 2005.

05-06 A. Meyer. A New Efficient Preconditioner for Crack Growth Problems. April 2005.

05-07 A. Meyer, P. Steinhorst. Überlegungen zur Parameterwahl im Bramble-Pasciak-CG fr gemischte FEM. April 2005.

05-08 T. Eibner, J. M. Melenk. Fast algorithms for setting up the stiffness matrix in hp-FEM: a comparison. June 2005.

05-09 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Vergleich der Fehlerindikatoren in Bezug auf die Netzsteuerung Teil I. June 2005.

05-10 A. Meyer, P. Nestler. Mindlin-Reissner-Platte: Vergleich der Fehlerindikatoren in Bezug auf die Netzsteuerung Teil II. July 2005.

05-11 A. Meyer, R. Unger. Subspace-cg-techniques for clinch-problems. September 2005.

05-12 P. Ciarlet, Jr, B. Jung, S. Kaddouri, S. Labrunie, J. Zou. The Fourier Singular Complement Method for the Poisson Problem. Part III: Implementation Issues. October 2005.

05-13 T. Eibner, J. M. Melenk. Multilevel preconditioning for the boundary concentrated *hp*-FEM. December 2005.

05-14 M. Jung, A. M. Matsokin, S. V. Nepomnyaschikh, Yu. A. Tkachov. Multilevel preconditioning operators on locally modified grids. December 2005.

05-15 S. Barrachina, P. Benner, E. S. Quintana-Ortí. Solving Large-Scale Generalized Algebraic Bernoulli Equations via the Matrix Sign Function. December 2005.

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/preprints.html`.

Preprintreihe des Chemnitzer SFB 393 – ISSN 1619-7178 (Print)