Open access • Proceedings Article • DOI:10.1145/566570.566590

Least squares conformal maps for automatic texture atlas generation

Bruno Levy, Sylvain Petitjean, Nicolas Ray, Jerome Maillot

Institutions: Centre national de la recherche scientifique, Alias Systems Corporation

Published on: 01 Jul 2002 - International Conference on Computer Graphics and Interactive Techniques

Topics: Texture atlas, Texture filtering, Texture mapping, Mesh parameterization and Similarity (geometry)

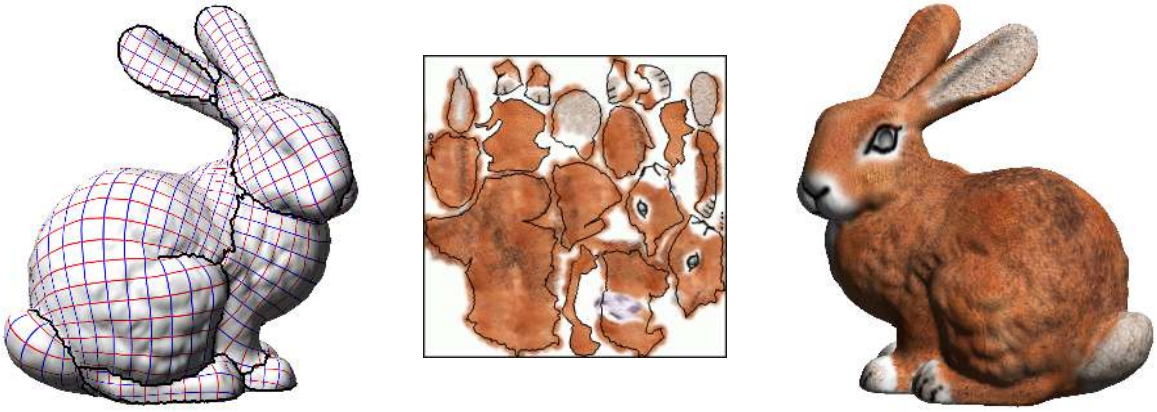
Least Squares Conformal Maps for Automatic Texture Atlas Generation

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, Jérôme Maillot

Least Squares Conformal Maps for Automatic Texture Atlas Generation

Bruno Lévy Sylvain Petitjean Nicolas Ray Jérome Maillot*

ISA (Inria Lorraine and CNRS), France



Abstract

A Texture Atlas is an efficient color representation for 3D Paint Systems. The model to be textured is decomposed into charts homeomorphic to discs, each chart is parameterized, and the unfolded charts are packed in texture space. Existing texture atlas methods for triangulated surfaces suffer from several limitations, requiring them to generate a large number of small charts with simple borders. The discontinuities between the charts cause artifacts, and make it difficult to paint large areas with regular patterns.

In this paper, our main contribution is a new quasi-conformal parameterization method, based on a least-squares approximation of the Cauchy-Riemann equations. The so-defined objective function minimizes angle deformations, and we prove the following properties: the minimum is unique, independent of a similarity in texture space, independent of the resolution of the mesh and cannot generate triangle flips. The function is numerically well behaved and can therefore be very efficiently minimized. Our approach is robust, and can parameterize large charts with complex borders.

We also introduce segmentation methods to decompose the model into charts with natural shapes, and a new packing algorithm to gather them in texture space. We demonstrate our approach applied to paint both scanned and modeled data sets.

CR Categories: I.3.3 [Computer Graphics] Picture/Image Generation; I.3.5 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing and texture; I.4.3 [Image processing]: Enhancement—Geometric Correction, Texture

Keywords: Texture Mapping, Paint Systems, Polygonal Modeling

*Alias|Wavefront

1 INTRODUCTION

A 3D paint system makes it possible to enhance the visual appearance of a 3D model by interactively adding details to it (colors, bump maps ...). If the discretization of the surface is fine enough, it is possible to directly paint its vertices [1]. However, in most cases, the desired precision for the colors is finer than the geometric details of the model. Assuming that the surface to be painted is provided with a parameterization, it is possible to use texture mapping to store colors in parameter space [9]. Parametric surfaces (such as NURBS) have a natural parameterization. For other representations, such as polygonal surfaces, finding a parameterization is non-trivial. To decorate polygonal models with regular patterns, the *lapped textures* approach [26] can be applied: local overlapping parameterizations are used to repeatedly map a small texture swatch onto a model.

A texture atlas is a more general representation (see, e.g., [13, 20, 23]). The model to be textured is partitioned into a set of parts homeomorphic to discs, referred to as *charts*, and each of them is provided with a parameterization. A texture atlas can be easily represented by standard file formats and displayed using standard texture mapping hardware. When used in a 3D paint system, a texture atlas should meet the following requirements:

- the chart boundaries should be chosen to minimize texture artifacts,
- the sampling of texture space should be as uniform as possible,
- the atlas should make an optimal use of texture space.

The generation of a texture atlas can be decomposed into the following steps:

1. **Segmentation:** The model is partitioned into a set of charts.
2. **Parameterization:** Each chart is 'unfolded', i.e. put in correspondence with a subset of $\mathbb{R}^2$.
3. **Packing:** The charts are gathered in texture space.

The remainder of this section presents the existing methods for these three steps, and their limitations with respect to the requirements mentioned above. We then introduce a new texture atlas generation method, meeting these requirements by creating charts with natural shapes, thus reducing texture artifacts.

1.1 Previous Work

Segmentation into charts. In [14] and [23], the model is interactively partitioned by the user. To perform automatic segmentation, Maillot *et al.* [20] group the facets by their normals. Several multi-resolution methods [7, 16] decompose the model into charts corresponding to the simplices of the base complex. In [27], Sander *et al.* use a region-growing approach to segmentation, merging charts according to both planarity and compactness criteria. All these approaches are designed to produce charts that can be treated by existing parameterization methods, which are limited to charts with convex borders. For this reason, a large number of charts is generated, which introduces many discontinuities when constructing a texture atlas.

Chart parameterization. Discrete Harmonic Map, described by Eck *et al.* [4], are the most widely used. They are approximations of Continuous Harmonic Maps [5], minimizing a *metric dispersion* criterion. Pinkal and Polthier [24] have shown the link between this criterion and another one named *conformality*, and have expressed both in terms of *Dirichlet energy*. Haker *et al.* [8] describe a similar method in the specific case of a surface triangulation homeomorphic to a sphere.

The theory on graph embedding has been studied by Tutte [30], where *Barycentric Maps* are introduced. The bijectivity of the so-defined parameterization is mathematically guaranteed. Floater [6] proposes specific weights improving the quality of the mapping, in terms of area deformations and conformality. In [18], a method is proposed to take additional constraints into account.

In all the methods mentioned above, since conformality is expressed as an indirect coupling between the parameters, boundary conditions are required, i.e. boundary nodes need to be fixed on a convex border in parameter space. Other expressions of conformality, such as the non-linear MIPS method [10], make it possible to overcome this problem, and let the boundary nodes be free to move. However, this latter method requires a time-consuming non-linear optimization, and may get stuck in a local minima of the non-linear function. In [12], Hurdal *et al.* propose a method based on *circle packings*, which are certain configurations of circles with specified pattern of tangencies known to provide a way to approximate a conformal mapping. Building circle packings is however quite expensive. The approach proposed in [28] consists in solving for the angles in parameter space. It results in a highly constrained optimization problem. Other methods [17, 25, 31] can also extrapolate the border, but do not guarantee the absence of triangle flips and require interaction with the user. We introduce here a conformal mapping method, offering more guarantees, efficiency and robustness than those approaches.

In the case of texture mapping, not only the bijectivity of the parameterization should be ensured, but also its ability to make an optimum use of texture memory, and to accurately represent a signal stored in texture space. Sander *et. al.* [27] describe an approach to minimize both a *texture stretch* criterion, and texture deviation between level of details. Since their approach is independent from the initial parameterization method, it can be applied to optimize the sampling of the parameterizations constructed by our method.

Charts packing in texture space. Finding the optimal packing of the charts in texture space is known as the *bin packing* problem. It has been studied by several authors, such as Milenkovic (see, e.g., [21]), but the resulting algorithms take a huge amount of time since the problem is NP-complete. To speed up these computations, several heuristics have been proposed in the computer graphics community. In the case of individual triangles, such a method is described by several authors (see, e.g., [3]). In the general case of charts, Sander *et al.* [27] propose an approach to pack the minimal area bounding rectangles of the charts. In our case, since the charts can have arbitrarily shaped borders, the bounding rectangle can be far away from the boundary of the charts. Therefore, a lot of texture space can be wasted. For this reason, we propose a more accurate packing algorithm that can handle the complex charts created by our segmentation and parameterization methods.

1.2 Overview

The paper is organized as follows. Since it is our main contribution, we will start by introducing Least Squares Conformal Maps (LSCMs), a new optimization-based parameterization method with the following properties (see Section 2 and Figure 1):

- Our criterion **minimizes angle deformations and non-uniform scalings**. It can be efficiently minimized by classical NA methods, and does not require a complex algorithm such as the ones used in [12] and in [28].
- We prove the **existence and uniqueness of the minimum** of this criterion. Therefore, the solver cannot get stuck in a local minimum, in contrast with non-linear methods [10, 25, 27, 31] where this property is not guaranteed.
- The borders of the charts do not need to be fixed, as with most of the existing methods [4, 6, 18]. Therefore, **large charts with arbitrarily shaped borders** can be parameterized.
- We prove that the orientation of the triangles is preserved, which means that **no triangle flip can occur**. However, as in [28], overlaps may appear, when the boundary of the surface self-intersects in texture space. Such configurations are automatically detected, and the concerned charts are subdivided. This problem was seldom encountered in our experiments (note that as with classical methods [4, 6], if all the border nodes are fixed on a convex polygon, no overlap can occur).
- We prove that the result is **independent of the resolution of the mesh**. This type of property may be usefull to reduce texture deviation when parameterizing different level of details of the same object.

In Section 3, we present a new segmentation method to decompose the model into charts. Thanks to the additional flexibility offered by LSCMs, it is possible to create large charts corresponding to meaningful geometric entities, such as biological features of characters and animals. The required number of charts is dramatically reduced, together with the artifacts caused by the discontinuities between the charts. Moreover, these large charts facilitate the use of regular patterns in a 3D paint system.

Section 4 presents our method to pack the charts in texture space. Since our segmentation method can create charts with complex borders, we pack the charts more accurately than with bounding rectangles, as in previous approaches. Our method is inspired by the strategy used by a 'Tetris' player. The paper concludes with some results, on both scanned and modeled meshes.

2 LEAST SQUARES CONFORMAL MAPS

In this section, we focus on the problem of parameterizing a chart homeomorphic to a disc. It will then be shown how to decompose the model into a set of charts, and how to pack these charts in texture space.

2.1 Notations

- scalars are denoted by normal characters x, y, u, v,
- vectors are denoted by bold characters $\mathbf{x} = (x, y)$,
- complex numbers are denoted by capitals $U = (u + iv)$,
- vectors of complex numbers are denoted by bold capitals $\mathbf{U}$,
- maps and matrices are denoted by cursive fonts $\mathcal{U}$, $\mathcal{X}$.

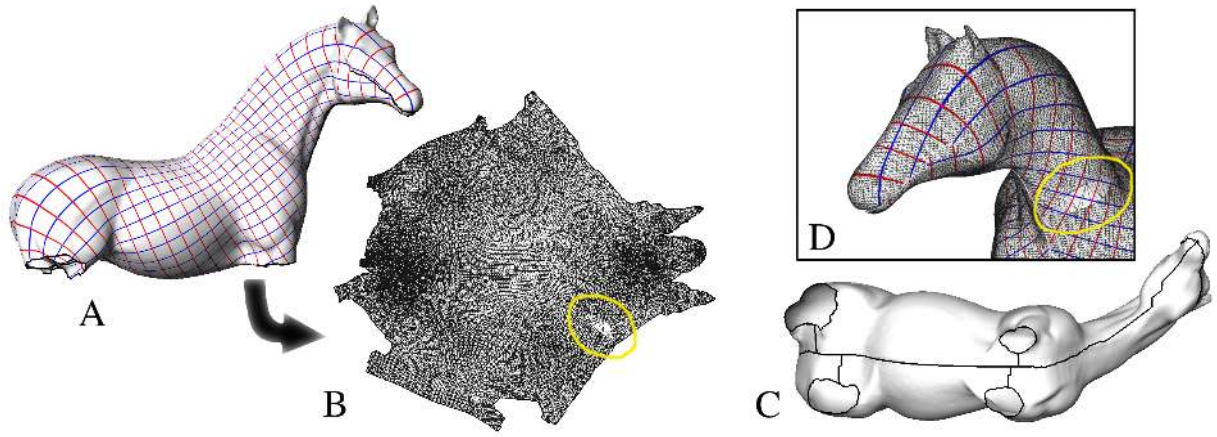


Figure 1: *The body of the scanned horse is a test case for the robustness of the method; it is a single very large chart of 72,438 triangles, with a complex border. A: Resulting iso-parameter curves; B: The corresponding unfolded surface, where the border has been automatically extrapolated; C: These cuts make the surface equivalent to a disc; D: The parameterization is robust, and not affected by the large triangles in the circled area (caused by shadow zones appearing during the scanning process).*

2.2 Conformal Maps

In this section, we quickly introduce the notion of conformal map. We will present further a new way to approximate the conformality criterion and the mathematical properties of this approximation.

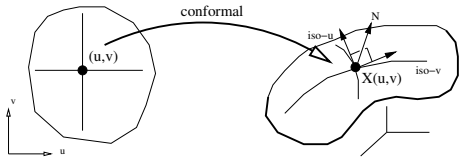


Figure 2: *In a conformal map, the tangent vectors to the iso-u and to the iso-v curves are orthogonal and have the same length.*

As shown in Figure 2, an application $\mathcal{X}$ mapping a (u, v) domain to a surface is said to be conformal if for each (u, v), the tangent vectors to the iso-u and iso-v curves passing through $\mathcal{X}(u, v)$ are orthogonal and have the same norm, which can be written as:

$$N(u,v) \quad \times \quad \frac{\partial \mathcal{X}}{\partial u}(u,v) \quad = \quad \frac{\partial \mathcal{X}}{\partial v}(u,v), \qquad (1)$$

where $N(u, v)$ denotes the unit normal to the surface. In other words, a conformal map is *locally isotropic*, i.e. maps an elementary circle of the (u, v) domain to an elementary circle of the surface.

It is possible to rewrite Equation 1 using differential operators, such as Laplace-Beltrami, as done in [24] and in [8], which results in the well known *cotangent* weighting coefficients (see e.g. [4]). The (u, v) parameters are then found to be the solution of two separate linear systems, one for u and one for v. The relation between u and v is **indirectly** taken into account by the right hand sides of the two systems. For this reason, this type of method requires the border to be fixed on a convex polygon. The MIPS method [10] does not have this restriction, and expresses conformality as a relation linking the coefficients of the metric tensor. However, the resulting equations are non-linear. Another approach has been described in [28], based on the remark that the criterion defining a conformal mapping should be independent of a translation, rotation and scaling in parameter space (i.e. a *similarity*). The unknowns are the angles at the corners of the triangles. This requires a time-consuming **constrained** optimization method.

Rather than discretizing the Laplace operator at the vertices of the triangulation, we instead take the dual path of considering the conformality condition on the triangles of the surface. Using the fact that a similarity can be represented by the product of complex numbers, we show how to turn the conformality problem into an **unconstrained** quadratic minimization problem. The u and v parameters are linked by a single global equation. This **direct** coupling of the u and v parameters makes it possible to efficiently parameterize large charts with complex borders, as shown in Figure 1. In this example, the cuts have been done manually (Figure 1-C), to create a large test case for the robustness of the method. (It will be shown in Section 3 how to automatically cut a model into charts homeomorphic to discs.)

Riemann's theorem states that for any surface S homeomorphic to a disc, it is possible to find a parameterization of the surface satisfying Equation 1. However, since we want to use the resulting parameterization for texture mapping, we add the constraint that the edges of the triangulation should be mapped to straight lines, and the mapping should vary linearly in each triangle. With this additional constraint, it is not always possible to satisfy the conformality condition. For this reason, we will minimize the violation of Riemann's condition in the least squares sense.

2.3 Conformality in a Triangulation

Consider now a triangulation $\mathcal{G} = \{[1 \dots n], \mathcal{T}, (\mathbf{p}_j)_{1 \leqslant j \leqslant n}\}$, where $[1 \dots n], n \geqslant 3$, corresponds to the vertices, where $\mathcal{T}$ is a set of n' triangles represented by triples of vertices, and where $\mathbf{p}_j \in \mathbb{R}^3$ denotes the geometric location at the vertex j. We suppose that each triangle is provided with a local orthonormal basis, where $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ are the coordinates of its vertices in this basis (i.e., the normal is along the z-axis). The local bases of two triangles sharing an edge are consistently oriented.

We now consider the restriction of $\mathcal{X}$ to a triangle T and apply the conformality criterion to the inverse map $\mathcal{U} : (x, y) \mapsto (u, v)$ (i.e. the coordinates of the points are given and we want their parameterization). In the local frame of the triangle, Equation 1 becomes

$$\frac{\partial \mathcal{X}}{\partial u} - i \frac{\partial \mathcal{X}}{\partial v} = 0,$$

where $\mathcal{X}$ has been written using complex numbers, i.e. $\mathcal{X} = x + iy$. By the theorem on the derivatives of inverse functions, this implies that

$$\frac{\partial \mathcal{U}}{\partial x} + i \frac{\partial \mathcal{U}}{\partial y} = 0, \qquad (2)$$

where $\mathcal{U} = u + iv$. (This is a concise formulation of the Cauchy-Riemann equations.)

Since this equation cannot in general be strictly enforced, we minimize the violation of the conformality condition in the least squares sense, which defines the criterion C:

$$C(T) = \int_T \left| \frac{\partial \mathcal{U}}{\partial x} + i \frac{\partial \mathcal{U}}{\partial y} \right|^2 dA = \left| \frac{\partial \mathcal{U}}{\partial x} + i \frac{\partial \mathcal{U}}{\partial y} \right|^2 A_T,$$

where A_T is the area of the triangle and the notation $|z|$ stands for the modulus of the complex number z.

Summing over the whole triangulation, the criterion to minimize is then

$$C(\mathcal{T}) = \sum_{T \in \mathcal{T}} C(T).$$

2.4 Gradient in a Triangle

Our goal is now to associate with each vertex j a complex number U_j such that the Cauchy-Riemann equation is satisfied (in the least squares sense) in each triangle. To this aim, let us rewrite the criterion $C(T)$, assuming the mapping $\mathcal{U}$ varies linearly in T.

We consider a triangle $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ of $\mathbb{R}^2$, with scalars u_1, u_2, u_3 associated with its vertices. We have:

$$\begin{pmatrix} \partial u/\partial x \\ \partial u/\partial y \end{pmatrix} = \frac{1}{d_T} \begin{pmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix},$$

where $d_T = (x_1 y_2 - y_1 x_2) + (x_2 y_3 - y_2 x_3) + (x_3 y_1 - y_3 x_1)$ is twice the area of the triangle.

The two components of the gradient can be gathered in a complex number:

$$\frac{\partial u}{\partial x} + i \frac{\partial u}{\partial y} = \frac{i}{d_T} \begin{pmatrix} W_1 & W_2 & W_3 \end{pmatrix} \begin{pmatrix} u_1 & u_2 & u_3 \end{pmatrix}^\top,$$

where

$$\left\{ \begin{array}{rcl} W_1 & = & (x_3 - x_2) + i(y_3 - y_2), \\ W_2 & = & (x_1 - x_3) + i(y_1 - y_3), \\ W_3 & = & (x_2 - x_1) + i(y_2 - y_1). \end{array} \right.$$

The Cauchy-Riemann equation (Equation 2) can be rewritten as follows:

$$\frac{\partial \mathcal{U}}{\partial x} + i \frac{\partial \mathcal{U}}{\partial y} = \frac{i}{d_T} \begin{pmatrix} W_1 & W_2 & W_3 \end{pmatrix} \begin{pmatrix} U_1 & U_2 & U_3 \end{pmatrix}^\top = 0,$$

where $U_j = u_j + i v_j$.

The objective function thus reduces to

$$C(\mathbf{U} = (U_1, \ldots, U_n)^\top) = \sum_{T \in \mathcal{T}} C(T), \quad \text{with}$$
$$C(T) = \frac{1}{d_T} \left| \begin{pmatrix} W_{j_1,T} & W_{j_2,T} & W_{j_3,T} \end{pmatrix} \begin{pmatrix} U_{j_1} & U_{j_2} & U_{j_3} \end{pmatrix}^\top \right|^2,$$

where triangle T has vertices indexed by j_1, j_2, j_3. (We have multiplied $C(T)$ by a factor of 2 to simplify the expression.)

2.5 Least Squares Conformal Maps

$C(\mathbf{U})$ is quadratic in the complex numbers $U_1, \ldots, U_n$, so can be written down as

$$C(\mathbf{U}) = \mathbf{U}^* \mathcal{C} \mathbf{U}, \tag{3}$$

where $\mathcal{C}$ is a Hermitian symmetric $n \times n$ matrix and the notation $\mathbf{U}^*$ stands for the Hermitian (complex) conjugate of $\mathbf{U}$. $\mathcal{C}$ is an instance of a Hermitian Gram matrix, i.e. it can be written as

$$\mathcal{C} = \mathcal{M}^* \mathcal{M},$$

where $\mathcal{M} = (m_{ij})$ is the sparse $n' \times n$ matrix (rows are indexed by triangles, columns are indexed by vertices) whose coefficient is

$$m_{ij} = \left\{ \begin{array}{ll} \frac{W_{j,T_i}}{\sqrt{d_{T_i}}} & \text{if vertex } j \text{ belongs to triangle } T_i, \\ 0 & \text{otherwise.} \end{array} \right.$$

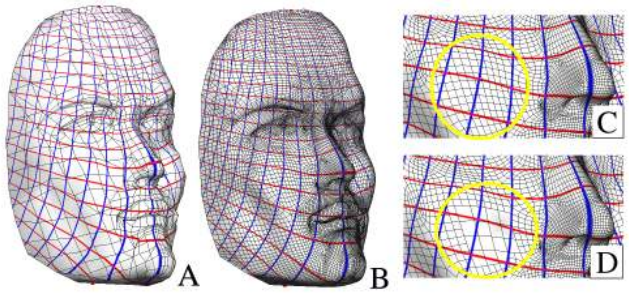


Figure 3: *Our LSCM parameterization is insensitive to the resolution of the mesh. The iso-parameter curves obtained on a coarse mesh (Figure A) and on a fine one (Figure B) are identical, and remain stable when the resolution varies within a mesh (circled zone in Figures C and D).*

For the optimization problem to have a non-trivial solution, some of the U_i's must be set to *a priori* values. Let us decompose the vector $\mathbf{U}$ as $(\mathbf{U}_f^\top, \mathbf{U}_p^\top)^\top$, where $\mathbf{U}_f$ is the vector of *free* coordinates of $\mathbf{U}$ (the variables of the optimization problem) and $\mathbf{U}_p$ is the vector of *pinned* coordinates of $\mathbf{U}$, of length p ($p \leqslant n$). Along the same lines, $\mathcal{M}$ can be decomposed in block matrices as

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_f & \mathcal{M}_p \end{pmatrix},$$

where $\mathcal{M}_f$ is a $n' \times (n-p)$ matrix and $\mathcal{M}_p$ is a $n' \times p$ matrix. Now, Equation 3 can be rewritten as

$$C(\mathbf{U}) = \mathbf{U}^* \mathcal{M}^* \mathcal{M} \mathbf{U} = \|\mathcal{M}\mathbf{U}\|^2 = \|\mathcal{M}_f \mathbf{U}_f + \mathcal{M}_p \mathbf{U}_p\|^2,$$

where the notation $\|\mathbf{v}\|^2$ stands for the inner product $< \mathbf{v}, \overline{\mathbf{v}} >$ ($\overline{\mathbf{v}}$ stands for the conjugate of $\mathbf{v}$).

Rewriting the objective function with only real matrices and vectors yields

$$C(\mathbf{x}) = \|\mathcal{A}\mathbf{x} - \mathbf{b}\|^2, \tag{4}$$

with

$$\mathcal{A} = \begin{pmatrix} \mathcal{M}_f^1 & -\mathcal{M}_f^2 \\ \mathcal{M}_f^2 & \mathcal{M}_f^1 \end{pmatrix}, \quad \mathbf{b} = -\begin{pmatrix} \mathcal{M}_p^1 & -\mathcal{M}_p^2 \\ \mathcal{M}_p^2 & \mathcal{M}_p^1 \end{pmatrix} \begin{pmatrix} \mathbf{U}_p^1 \\ \mathbf{U}_p^2 \end{pmatrix},$$

where the superscripts ^{1} and ^{2} stand respectively for the real and imaginary part, $\|\mathbf{v}\|$ stands this time for the traditional L_2-norm of a vector with real coordinates and $\mathbf{x} = (\mathbf{U}_f^{1\top}, \mathbf{U}_f^{2\top})^\top$ is the vector of unknowns.

Note that $\mathcal{A}$ is a $2n' \times 2(n-p)$ matrix, $\mathbf{b}$ is a vector of $\mathbb{R}^{2n'}$ and $\mathbf{x}$ is a vector of $\mathbb{R}^{2(n-p)}$ (the u_i and v_i coordinates of the vertices in parameter space that are allowed to move freely).

2.6 Properties

The above minimization problem has several fundamental properties which are proved in the appendix:

- The matrix $\mathcal{A}$ has full rank when the number of pinned vertices, i.e. p, is larger than or equal to 2.
- As a consequence, the minimization problem has a unique solution when $p \geqslant 2$, given by $\mathbf{x} = (\mathcal{A}^\top \mathcal{A})^{-1} \mathcal{A}^\top \mathbf{b}$. The best value for p is 2, since in this case the mapping $\mathcal{U}$ can be fully conformal if the surface is developable (i.e. the minimum of the objective function is zero). In our experiments, we have pinned the two vertices maximizing the length of the shorted path between them (i.e. the graph diameter).
- The solution to the minimization problem is invariant by a similarity in texture space.
- The solution to the minimization problem is independent of the resolution of the mesh. This property is illustrated in Figure 3.
- In texture space, all the triangles are consistently oriented if the pinned vertices are chosen on the boundary of $\mathcal{T}$. In other words, triangle flips cannot occur.

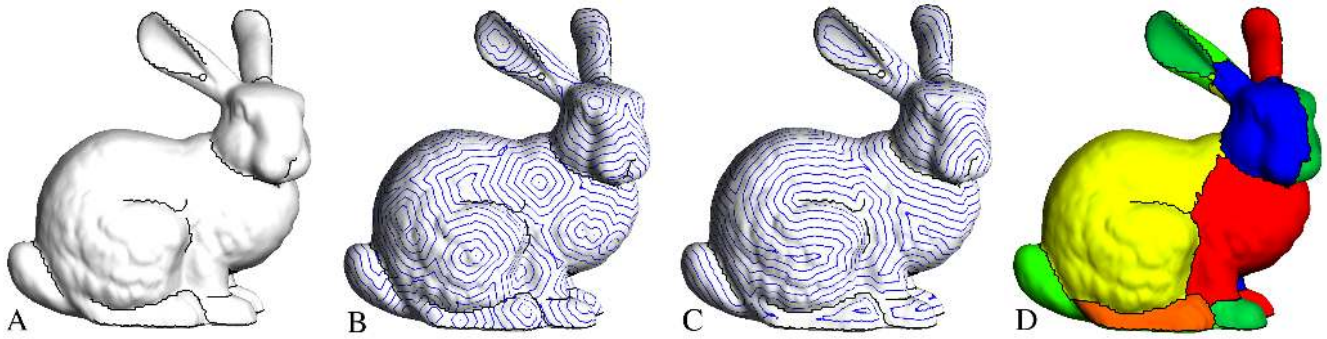


Figure 4: *A: Result of the features detection algorithm; B: The $distance\_to\_seed$ function is not an optimal choice for driving our chart growing process; C: The $distance\_to\_features$ function shows iso-contours with more natural shapes; D: Result of our segmentation algorithm, driven by the $distance\_to\_features$ function.*

3 SEGMENTATION

The segmentation algorithm decomposes the model into a set of charts. The design of the algorithm aims at meeting the following two requirements:

1. charts boundaries should be positioned in such a way that most of the discontinuities between the charts will be located in zones where they will not cause texture artifacts,
2. charts must be homeomorphic to discs, and it must be possible to parameterize them without introducing too much deformation.

For the first point, since the shading models depend on the normal, zones of high curvatures cause sharp variations of lighting. In these zones, a texture artifact will not be noticeable, since it will be negligible compared to the shading variation. Thus, to minimize artifacts, we will design the segmentation algorithm in such a way as to avoid chart boundaries in flat zones. In other words, it is suitable to generate large charts with most of their boundaries in high curvature zones.

For the second point, we will present an automatic approach, mimicking the way a user manually segments a model. Basically, the user attempts to decompose the model into parts resembling cylinders. To detect such cylinders, we will use an approach inspired by Morse theory, characterizing a function defined over the surface (see, e.g., [29]).

The next two sections present a feature detection algorithm, which finds curves corresponding to high curvature zones of the model, and a chart growing algorithm making them meet at these feature curves. Then, it will be shown how to validate the result, and subdivide the charts if needed. In the remainder of this section, we suppose that the surface is represented by a halfedge based data structure (see, e.g., [19]).

3.1 Detect Features

The features detection phase can be outlined as follows:

1. Compute a sharpness criterion on the edges. We use here the second order differences (SOD), i.e. the angle between the normals, as in [11]. It is also possible to use more elaborate criteria.
2. Choose a threshold τ so that a certain proportion of the edges is filtered out. In our examples, we kept 5 percent of the detected edges.
3. For each of the remaining edges, grow a feature curve by applying Algorithm 1.

Algorithm 1 attempts to anticipate the best paths, and filters out the small features caused by noise. Tagging the neighborhoods of the detected features avoids generating a large number of features in zones of high curvature. In our examples, the parameters are set as follows: $max\_string\_length = 5$, which controls the size of the discontinuities to be filled, and $min\_feature\_length = 15$.

expand_feature_curve(halfedge $start$)
  vector<halfedge> $detected\_feature$
  for halfedge $h \in \{ start$, opposite($start$) $\}$
    halfedge $h' \leftarrow h$
    do
      use depth-first search to find the string S of halfedges
      starting with h' and such that:
        • two consecutive halfedges of S share a vertex
        • the length of S is $\leqslant$ than max_string_length
        • $sharpness(S) \leftarrow \sum_{e \in S} sharpness(e)$ is maximum
        • no halfedge of S goes backward (relative to h')
        • no halfedge of S is tagged as a feature neighbor
      $h' \leftarrow$ second item of S
      append h' to $detected\_feature$
    while($sharpness(S) > max\_string\_length \times \tau$)
  end // *for*
  if (length($detected\_feature$) > $min\_feature\_length$) **then**
    tag the elements of $detected\_feature$ as features
    tag the halfedges in the neighborhood of $detected\_feature$
      as feature neighbors
  end // *if*
end // *expand_feature_curve*

Algorithm 1: Features growing

3.2 Expand Charts

Once the sharp features have been detected, the charts can be created. Our method is a greedy algorithm, expanding all the charts simultaneously from a set of *seeds*. It is similar to the s-source Dijkstra algorithm used in [4] and to the region-growing paradigm used in computer vision. Since we want chart boundaries to meet at the level of features, the s-source algorithm is modified as follows:

- To select the set of seeds, the intuitive idea is to 'reverse engineer' the expected result. More precisely, we use the following method: a front is propagated from the borders and the feature curves detected by the previous algorithm, to compute a $distance\_to\_features$ function at each facet. Then, the seeds are found to be the local maxima of this $distance\_to\_features$ function.
- For closed surfaces without any detected feature, propagation is initialized from the two extremities of a diameter of the facets graph, as done in [15].
- Our s-source propagation uses $-distance\_to\_features$ as the priority function, rather than $distance\_to\_seeds$. The advantage of this approach is shown in Figure 4.

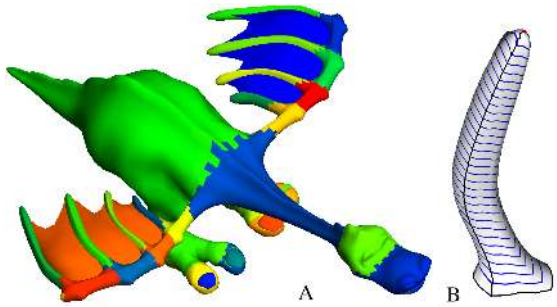


Figure 5: *A: Our segmentation algorithm detects cylindrical shapes; B: An additional cut is added to 'sock-shaped' extremal cylinders.*

expand_charts
priority_queue<halfedge> $Heap$ sorted by $dist(facet(halfedge))$
set<edge> $chart\_boundaries$ initialized with all the edges of the surface
// Initialize Heap
foreach facet F where dist(F) is a local maximum
create a new chart with seed F
add the halfedges of F to $Heap$
end *// foreach*

// Charts-growing phase
while($Heap$ is not empty)
halfedge $h \leftarrow e \in Heap$ such that dist(e) is maximum
remove h from $Heap$
facet $F \leftarrow$ facet(h)
facet $F_{opp} \leftarrow$ the opposite facet of F relative to h
if (chart(F_{opp}) is undefined) **then**
add F_{opp} to chart(F)
remove E from $chart\_boundaries$
remove non-extremal edges from $chart\_boundaries$,
// (i.e. edges that do not link two other chart boundary edges)
add the halfedges of F_{opp} belonging to
$chart\_boundaries$ to $Heap$
elseif (chart(F_{opp}) $\neq$ chart(F) **and**
max_dist(chart(F)) - dist(F) $< \varepsilon$ **and**
max_dist(chart(F_{opp})) - dist(F) $< \varepsilon$) **then**
merge chart(F) and chart(F_{opp})
end *// if*
end *// while*
end *// expand_charts*

Algorithm 2: Charts growing.

- Charts are merged if they meet at a small distance $d < \varepsilon$ from their seed. In our experiments, $\varepsilon = maxdist/4$, where $maxdist$ denotes the global maximum of $distance\_to\_features$.

Our charts growing algorithm (Algorithm 2) uses the following data:

- $distance\_to\_feature$ is stored in each facet F, and denoted $dist(F)$;
- for each chart C, the scalar $max\_dist(C)$ denotes the maximum distance to features for all the facets of C;
- The set of edges $chart\_boundaries$ represents the borders of all charts. It makes it possible for a patch to be its own neighbor while remaining a topological disc.

As shown in Figure 5, our algorithm can detect cylindrical shapes, as the approach proposed in [15]. In our case, two configurations can be distinguished:

- The cylinder corresponds to an extremity, such as the 'fingers' of the dinosaur's wings (Figure 5-B). The corresponding charts have the shape of a sock. This configuration is detected by computing the area/perimeter ratio. In this case, to facilitate the parameterization phase, a cut is added by starting from the seed and cutting the edges along the steepest-descent path.
- The cylinder is non-extremal, and therefore non-capped. Algorithm 2 generates suitable boundaries, without requiring any special treatment (the resulting chart is 'rolled' around the cylinder).

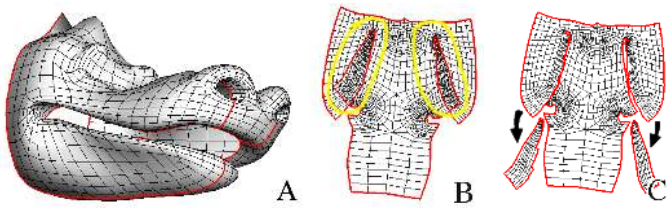


Figure 6: *A: The dinosaur's head made of a single chart; B: Despite the absence of triangle flips, overlaps may occur, caused by self-intersections of the border; C: They can be removed by subdividing the chart.*

3.3 Validate Charts

The so-constructed charts are then parameterized using the method presented in Section 2. After that, the following two criteria are tested:

- As mentioned in Section 2, no triangle flip can occur, and the border can be extrapolated. However, since the border can be non-convex, a new class of overlaps can be encountered. They are caused by a self-intersection of the border, as in [28]. Such configurations can be efficiently detected by the hardware, by drawing the parameter space in stencil mode. The stencil pixels drawn more than once correspond to overlaps. If such overlaps are detected, the corresponding chart is subdivided, by cutting it along the edges on the border of the overlapped zone, as shown in Figure 6. Note that our segmentation algorithm would not generate a single chart for the dinosaur's head (see Figure 5). In practice, the overlap problem has seldom appeared in our experiments, and was caused by tiny loops formed by the border.
- Our criterion respects angles very well, as shown in the results section. As far as areas are concerned, large charts with zones of high curvature may result in large area variations. To detect these problems, the minimum and maximum model area/texture area ratio is measured over the facets. If the max/min ratio is greater than a certain threshold, the concerned chart is split, by growing two charts from the facets corresponding to the minimum and the maximum. In the examples shown here, the threshold has been set to 2.

4 PACKING

Once the model is decomposed into a set of parameterized charts, it is possible to create a texture atlas by merging all the (u, v) domains of the charts. Usually, only a limited amount of texture memory is available. It is then suitable to minimize the unused space. In other words, given a set of possibly non-convex polygons, we want to find a non-overlapping placement of the polygons in such a way that the enclosing rectangle is of minimum area. The so-obtained texture coordinates are then re-scaled to fit the size of the texture.

The packing problem is known to be NP-complete (see, e.g., [21] and [22]). Approaches based on computational geometry show good performances in terms of minimization of lost area, but are not efficient enough for large and complex data sets. For this reason, several heuristics have been proposed in computer graphics. For instance, in the method proposed by Sander *et al.* [27], the bounding rectangles of the charts are packed. In our case, since

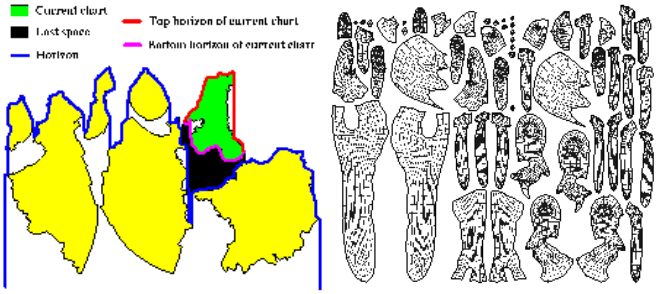


Figure 7: *A: Our packing algorithm inserts the charts one by one, and maintains the 'horizon' (in blue) during the process. Each chart (in green) is inserted at the position minimizing the 'wasted space' (in black) between its bottom horizon (in pink) and the current horizon. The top horizon (in red) of the current chart is then used to update the horizon. B: Result on the dinosaur data set.*

the border may have an arbitrary shape, the bounding rectangle is not an accurate approximation.

For this reason, we propose a different algorithm, that packs the charts directly rather than their bounding rectangles. As in [2], our algorithm is inspired by how a 'Tetris' player would operate, but without approximating the charts by their bounding boxes:

1. Each chart is rescaled to make its area in (u, v) space equal to its area in (x, y, z) space.
2. The maximum diameter of the charts are oriented vertically and sorted in decreasing order.
3. As shown in Figure 7, for each chart C, the top horizon $h_C^\top$ (in red) and bottom horizon $h_C^\perp$ (in pink) is computed. Rather than being represented by their bounding rectangles, the charts are approximated by the area between the two curves $h_C^\top$ and $h_C^\perp$. As in classical approaches, in order to avoid unwanted blends caused by mip-mapping, an additional margin is added to the horizons.
4. The charts are inserted one by one, using the method described below.

As shown in Figure 7-A, the piecewise linear function $h(u)$ representing the 'horizon' is maintained by the algorithm. For each chart C, the u_C coordinate at the lower left corner of C is chosen in such a way that the lost space (in black) between the bottom horizon $h_C^\perp$ of C (in pink) and the current horizon h (in blue) is minimized (see Figure 7-A). Then, the horizon h is updated using the top horizon $h_C^\top$ of the current chart (in red). Since the parameter space is discretized into texels, it seems natural to represent the horizons by arrays of discretized values, with texture resolution. This makes the algorithm much simpler than using piecewise linear functions. For a chart C, all discrete values for u_C are tested. The algorithm performs well, and takes less than one second to process all the data sets we have tested.

5 RESULTS

We have applied our method to different data sets, comprising meshes created with a 3D modeler (using subdivision surfaces) and scanned meshes. The results are equivalent to those obtained with MIPS [10], but with mathematical guarantees, and can be more efficiently computed. As shown in Table 1, the stretch (see [27]) measured on the result is of the same order as when using standard methods (e.g. [4, 6]). To optimize the mapping, it is easy to post-process the result of our method by the algorithm proposed in [27]. Since the border nodes are naturally positionned (rather than arbitrarily fixed on a convex polygon), the result can be better (see Table 1).

Figure 8 shows some texture atlases. Note the presence of small charts, most of them corresponding to geometric details of the models (teeth, hoofs ...). This is not a problem for most paint systems,

	Harmonic Maps	LSCM
stretch (before optim.)	3.2	3.5
stretch (after optim.)	1.65	1.52

Table 1: *Stretch optimization of the 'cow head' data set*

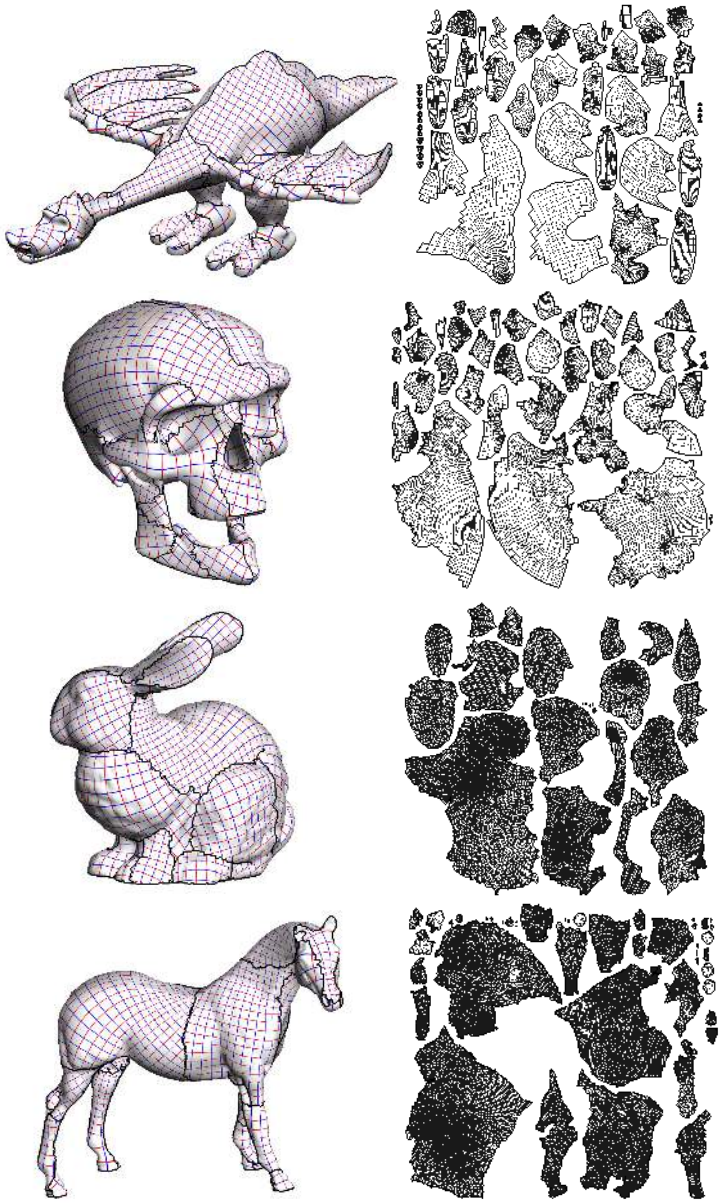


Figure 8: *Data sets and associated texture space constructed by our method.*

	dinosaur	skull	bunny	horse
♯ vertices	14,669	16,949	34,834	48,485
♯ facets	14,384	15,124	69,451	96,968
♯ charts	43	40	23	44
segmentation time (s)	8	17	30	43
parameterization time (s)	10	23	95	190
packing ratio (rectangles)	0.48	0.51	0.43	0.37
packing ratio (our algo.)	0.55	0.55	0.6	0.58
stretch (before optim.)	2.9	2.5	1.16	1.14
stretch (after optim.)	1.26	1.55	1.14	1.12

Table 2: *Statistics and timings.*

Figure 10: *Hand-painted 3D models. Our LSCM method facilitates the use of procedural textures and complex patterns.*

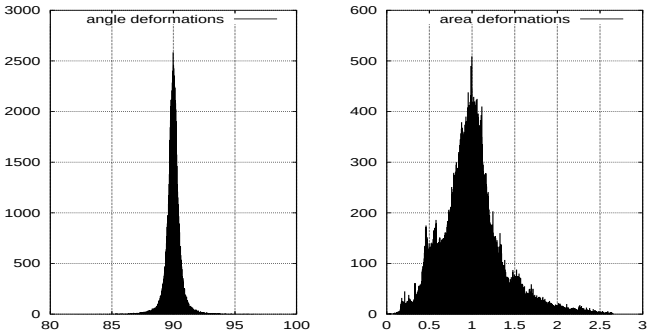


Figure 9: *Angle and area deformations histograms ('Horse' data set).*

that can treat them properly. Some examples of textured models are shown in Figure 10. Table 2 shows the sizes of the data sets, the number of created charts, and the following statistics, obtained on a 1.3 GHz Pentium III (note that the timings for the packing algorithm are not included, since they are negligible):

- *time to segment the model into charts*;
- *time to parameterize the charts*. Our LSCM criterion (Equation 4) is minimized using the CG (Conjugate Gradient) algorithm. The independence to resolution suggests that a multi-grid approach would be even more efficient;
- *packing ratio* obtained using an enclosing rectangle packing approach [27] and our algorithm.
- *stretch* measured before and after applying Sander *et. al.*'s optimization method as a post-processing (see [27]).

The left histogram in Figure 9 shows the distribution of the angles in degrees between u and v gradient vectors. The mapping is nearly conformal in each triangle (the differences of lengthes between the u and v gradients we have measured are very near to zero). The right histogram shows the area deformations obtained with the 'Horse' data set, before stretch optimization. This histogram, showing texture area/model area ratios has been normalized, i.e. scaled in such a way that the mean value is mapped to 1. Note that since the mapping is nearly isotropic in each triangle, the L^2 and L^∞ stretch histogram (not shown here) have exactly the same appearance as the area histogram. As can be seen, even though our LSCM criterion is not designed to punish area deformations, few facets are distorted, and can easily be fixed by post-processing using Sander *et. al.*'s method. The resulting texture atlases combine the advantages of LSCM (few chart discontinuities) and stretch-optimized parameterization (uniform sampling).

CONCLUSION

In this paper, we have presented a new automatic texture atlas generation method for polygonal models. Overall, we have proposed a complete and mathematically valid solution to the parameterization of complex models which proved to be more efficient and robust than existing methods and available tools in real production environments. Our segmentation algorithm, driven by detected features and inspired by Morse theory, decomposes the model into charts with natural shapes, corresponding to meaningful geometric entities. These two algorithms may have applications in other domains, such as re-meshing and data compression. We have successfully applied our technique to both scanned and synthetic data sets, making it possible to use existing 3D paint systems with them (DeepPaint3D, Painter). In future works, to parameterize huge models, we will consider out-of-core algorithm, and analyze different numerical methods to minimize the LSCM criterion, including multigrid approaches and pre-conditioned CG. Including the stretch criterion directly into the LSCM criterion is also another possible future direction of research.

ACKNOWLEDGMENTS

We want to thank the Graphite development team (`http://www.loria.fr/~levy/graphite`), especially Ben Li and Bijendra Vishal. Thanks also to the reviewers for their comments and help in improving this paper.

A PROPERTIES OF LSCMS

The minimization problem of Section 2 has several interesting properties when the number p of pinned vertices in parameter space is sufficient. In what follows, $\mathcal{T}$ is assumed to be homeomorphic to a disc.

A.1 Full Rank

We first show that the matrices $\mathcal{M}_f$ and $\mathcal{A}$ have full rank when $p \geqslant 2$ (p denotes the number of pinned vertices).

For this, recall that a triangulation that is topologically a disc can be incrementally constructed with only two operations (cf. Figure 11): the *glue* operation creates one new vertex and one new face, and the *join* operation creates one new face. Thus, incremental construction creates at most as much vertices as faces. Since the simplest triangulation (one triangle) has one face and three vertices, we have that $n' \geqslant n - 2$ (where n denotes the number of vertices, and n' the number of triangles, as in the rest of the paper).

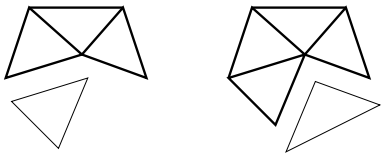


Figure 11: *Incremental construction of a triangulation that is topologically a disc. Left: glue two triangles along an edge. Right: join two existing vertices, creating a new triangle.*

We first show that the rank of $\mathcal{M}_f$ is $n-p$ when $p \geqslant 2$. First note that since $n' \geqslant n-2$, $\min(n', n-p) = n-p$ if $p \geqslant 2$ and the rank of $\mathcal{M}_f$ is at most $n-p$. We assume that $\mathcal{T}$ is incrementally constructed with glue and join operations and prove the result by induction on the size of $\mathcal{M}_f$. We also assume, without loss of generality, that the p pinned vertices are concentrated in the initial triangulation. Let $n_i', n_i - p$ be the dimensions of the matrix $\mathcal{M}_f^{(i)}$ at step i. Observe that since $\mathcal{T}$ is a non-degenerate triangulation, none of the coefficients W_{j,T_i} is zero.

At step 0, the triangulation has $n_0 - p = 1$ vertices and $n_0' \geqslant 1$ triangles. $\mathcal{M}_f^{(0)}$ has a single column and, since $\mathcal{T}$ is a proper triangulation, some of its coefficients are non-zero and it has rank $1 = n_0 - p$.

Assume that the property holds after step i. If step $i+1$ is a join, then the number of rows of $\mathcal{M}_f^{(i)}$ grows by 1 while the number of columns is unchanged, so the rank is $n_{i+1} - p = n_i - p$. If step $i+1$ is a glue, a new vertex $\mathbf{v}_{i+1}$ and a new triangle T are added. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be the other vertices of T. The new matrix $\mathcal{M}_f^{(i+1)}$ is as follows:

$$\left(\begin{array}{ccc:c} & \mathcal{M}_f^{(i)} & & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \hdashline \frac{W_{1,T}}{\sqrt{d_T}} & \frac{W_{2,T}}{\sqrt{d_T}} & 0 \cdots 0 & \frac{W_{i+1,T}}{\sqrt{d_T}} \end{array}\right).$$

It is now easy to see that its columns are linearly independent. Indeed, assume there are complex numbers λ_j such that

$$\sum_{j=1}^{n_{i+1}} \lambda_j \mathbf{m}_j^{(i+1)} = 0, \tag{5}$$

where the $\mathbf{m}_j^{(i+1)}$ are the column vectors of $\mathcal{M}_f^{(i+1)}$. If we look at the first n_i' coordinates of the column vectors, then Equation 5 reduces to $\sum_{j=1}^{n_i} \lambda_j \mathbf{m}_j^{(i)} = 0$, which implies that $\lambda_j = 0, j = 1, \ldots, n_i$, since $\mathcal{M}_f^{(i)}$ has full rank. Now the equation linking the last coordinate of the vectors $\mathbf{m}_j^{(i+1)}$ reduces to $\lambda_{n_{i+1}} W_{i+1,T}/\sqrt{d_T} = 0$, implying that $\lambda_{n_{i+1}} = 0$. Thus the columns of $\mathcal{M}_f^{(i+1)}$ are linearly independent and the matrix has full rank. The result is proved.

Since $\mathcal{M}_f$ has rank $n-p$, both $\mathcal{M}_f^1$ and $\mathcal{M}_f^2$ have rank $n-p$ when $p \geqslant 2$. In turn, this implies that $\mathcal{A}$ has rank $2(n-p)$ when $p \geqslant 2$.

A.2 Single Minimum

We now show that, when $p \geqslant 2$, $C(\mathbf{U})$ has a unique minimum.

First, notice that

$$\frac{\partial C}{\partial \mathbf{x}} = 2(\mathcal{A}^\top \mathcal{A}\mathbf{x} - \mathcal{A}^\top \mathbf{b}).$$

Now, since the rank of the Gram matrix of $\mathcal{A}$ (i.e. $\mathcal{A}^\top\mathcal{A}$) is the same as the rank of $\mathcal{A}$, $\mathcal{A}^\top\mathcal{A}$ has rank $2(n-p)$ when $p \geqslant 2$. Since $\mathcal{A}^\top\mathcal{A}$ is a square $2(n-p) \times 2(n-p)$ matrix, it is thus invertible and the minimization problem has a unique solution (when $p \geqslant 2$)

$$\mathbf{x} = (\mathcal{A}^\top\mathcal{A})^{-1}\mathcal{A}^\top\mathbf{b}.$$

The minimum of $C(\mathbf{U})$ is zero when $\mathcal{A}\mathbf{x} = \mathbf{b}$, i.e. when $\mathcal{A}$ is invertible. Since it has full rank, this happens exactly when $\mathcal{A}$ is square, i.e. when $n' = n-p$. Using the fact that $n' \geqslant n-2$, this implies that $p = 2$. We conclude that the mapping $\mathcal{U}$ is fully conformal (barring self-intersections) exactly when $p = 2$ and the triangulation $\mathcal{T}$ is built only with glue operations.

A.3 Invariance by Similarity

We now prove that if $\mathbf{U}$ is a solution to the minimization problem, then $z\mathbf{U} + \mathbf{T}$ is also a solution, for all $z \in \mathbb{C}$ and $\mathbf{T} = (z', \ldots, z'), z' \in \mathbb{C}$. In other words, the problem is invariant by a similarity transformation.

First note that the vector $\mathbf{H} = (1, \ldots, 1)^\top$ is trivially in the kernel of $\mathcal{M}$, since $W_1 + W_2 + W_3 = 0$ in each triangle. Assume $\mathbf{U}$ is a solution of the problem. We get:

$$\begin{aligned} C(z\mathbf{U} + \mathbf{T}) &= z\overline{z}\, C(\mathbf{U}) + 2z\mathbf{T}^*\mathcal{C}\mathbf{U}, \\ &= z\overline{z}\, C(\mathbf{U}) + 2z(\mathcal{M}\mathbf{T})^*\mathcal{M}\mathbf{U} = z\overline{z}\, C(\mathbf{U}), \end{aligned}$$

because $\mathbf{T} = z'\mathbf{H}$ is in the kernel of $\mathbf{M}$. If $C(\mathbf{U}) = 0$, then $C(z\mathbf{U} + \mathbf{T}) = 0$.

A.4 Independence to Resolution

We now show that if a given mesh is 'densified', then the solution to the augmented optimization problem restricted to the vertices of the initial mesh is the same. We prove this result when a single triangle T is split into three triangles, but the proof generalizes easily to a more general setting. So let $\mathbf{v}$ be the new vertex introduced in triangle T, i.e. as a linear combination of vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$:

$$\mathbf{v} = \sum_{i=1}^{3} \alpha_i \mathbf{v}_i, \quad \sum_{i=1}^{3} \alpha_i = 1, \quad \alpha_i > 0.$$

Assume also for the sake of simplicity that none of $\mathbf{v}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ is pinned. Call T_i $(i = 1, \ldots, 3)$ the triangle created that does not have $\mathbf{v}_i$ as vertex. Then it is easy to see that $d_{T_i} = \alpha_i d_T$.

$\mathcal{M}_f$ is an $n' \times (n-p)$ matrix. After insertion of $\mathbf{v}$, the new matrix $\mathcal{M}_f^+$ is $(n'+2) \times (n+1-p)$. Indeed, one vertex is added, augmenting the number of columns by one, and three new triangles replace an old one, augmenting the number of rows by two. The structure of these matrices is as follows:

$$\mathcal{M}_f = \left(\begin{array}{c} \mathcal{N}_f \\ \hdashline \mathcal{F} \quad 0 \cdots 0 \end{array}\right), \quad \mathcal{M}_f^+ = \left(\begin{array}{cc:c} & \mathcal{N}_f & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \hdashline \mathcal{L} & \mathbf{0} & \mathcal{P} \end{array}\right).$$

where $\mathcal{N}_f$ is an $(n'-1) \times (n-p)$ matrix, $\mathcal{F}$ is 1×3, $\mathcal{L}$ is 3×3 and $\mathcal{P}$ is 3×1. If the coefficients of $\mathcal{F}$ are denoted by $f_j = W_{j,T}/\sqrt{d_T}$, then it is easy to observe that the coefficients of $\mathcal{L} = (l_{ij})$ and $\mathcal{P} = (p_i)$ satisfy

$$l_{ij} = \frac{1}{\sqrt{\alpha_i}}\big(\alpha_i f_j - \alpha_j f_i\big), \quad p_i = \frac{1}{\sqrt{\alpha_i}} f_i. \tag{6}$$

The $(n-p) \times 1$ solution to the initial problem is:

$$\mathbf{U}_f = (\mathcal{M}_f^*\mathcal{M}_f)^{(-1)}\mathcal{M}_f^*\mathcal{M}_p\mathbf{U}_p.$$

Consider the $(n+1-p)\times 1$ solution to the augmented problem:

$$\mathbf{U}_f^+ = (\mathcal{M}_f^{+*}\mathcal{M}_f^+)^{(-1)}\mathcal{M}_f^{+*}\mathcal{M}_p^+\mathbf{U}_p^+. \quad (7)$$

Using the relations of Equation 6 and the fact that $f_1+f_2+f_3=0$, it suffices then to observe that $\mathbf{U}_f^+ = (\mathbf{U}_f^\top, U_\mathbf{v})^\top$ is the (unique) solution to 7, where

$$U_\mathbf{v} = \alpha_1 U_1 + \alpha_2 U_2 + \alpha_3 U_3.$$

In other words, the least squares conformal parameterization is unchanged at the old vertices and is the barycenter of the parameterizations of $\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3$ at the new vertex $\mathbf{v}$.

A.5 Preserving Orientations

We now sketch the proof that least squares conformal maps preserve orientations, i.e. there are no triangle flips.

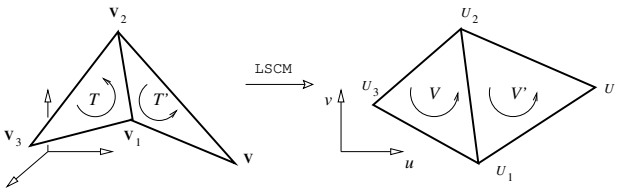


Figure 12: *LSCMs preserve orientations.*

As a preliminary, note first that if complex numbers W_i are associated to vertices of a triangle as in Section 2.4, with vertices ordered counterclockwise, then

$$\zeta_T = i\big(\overline{W_2}W_1 - \overline{W_1}W_2\big) \quad (8)$$

is positive (and equal to $2d_T$).

We again assume that the triangulation $\mathcal{T}$ is incrementally constructed with the glue and join operations. Denote the current triangulation by $\mathcal{T}_i$. For the join operation, the result is trivial. We now prove the result when the current step is a glue. We use the notations of Figure 12. Let V and V' be the images of T and T' in parameter space. Let also W_j (resp. W_j') be complex numbers attached to T (resp. T') and X_j (resp. X_j') be complex numbers attached to V (resp. V'). Since the local bases of two triangles of $\mathcal{T}$ sharing an edge are consistently oriented, both ζ_T – as defined in Equation 8 – and

$$\zeta_{T'} = i\big(\overline{W_1'}W_2' - \overline{W_2'}W_1'\big)$$

are positive. If we assume that the unfolding of $\mathcal{T}_i$ has no triangle flips, then we also have that $\zeta_V > 0$, where ζ_V is defined as in Equation 8, replacing W_j by X_j.

Now, writing down the equations defining $\mathbf{U}_f^+ = (\mathbf{U}_f^\top, U_\mathbf{v})^\top$ as in the previous section, we find that

$$W_1'U_1 + W_2'U_2 + W_\mathbf{v}'U_\mathbf{v} = 0, \quad (9)$$

where $U_1, U_2, U_\mathbf{v}$ are the parameterizations of vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}$ (U_1, U_2 being unchanged by addition of $\mathbf{v}$). Using the fact that $W_1' + W_2' + W_3' = 0, X_1' = U_2 - U_\mathbf{v}$ and $X_2' = U_\mathbf{v} - U_1$, Equation 9 rewrites as

$$W_2'X_1' - W_1'X_2' = 0. \quad (10)$$

Using Equation 10 and the definition in Equation 8, we have:

$$\begin{aligned}\zeta_{V'} &= i\big(\overline{X_1'}X_2' - \overline{X_2'}X_1'\big),\\ &= \frac{X_1'\overline{X_1'}}{W_1'\overline{W_1'}}i\big(\overline{W_1'}W_2' - \overline{W_2'}W_1'\big) = \frac{X_1'\overline{X_1'}}{W_1'\overline{W_1'}}\zeta_{T'} > 0.\end{aligned}$$

Thus, V' is consistently oriented and the glue operation does not produce a triangle flip, proving the result.

References

[1] M. Agrawala, A. Beers, and M. Levoy. 3D painting on scanned surfaces. In *Proc. 1995 Symposium on Interactive 3D Graphics*, 1995.

[2] Y. Azar and L. Epstein. On 2d packing. *J. of Algorithms*, (25):290–310, 1997.

[3] P. Cigogni, C. Montani, C. Rocchini, and R. Scopino. A general method for recovering attributes values on simplified meshes. In *Proc. of IEEE Visualization Conf.*, pages 59–66. ACM Press, 1998.

[4] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Conf. Proc.*, pages 173–182. Addison Wesley, 1995.

[5] J. Eells and L. Lemaire. Another report on harmonic maps. *Bull. London Math. Soc.*, 20:385–524, 1988.

[6] M. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, April 1997.

[7] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *SIGGRAPH 00 Conf. Proc.*, pages 95–102. ACM Press, 2000.

[8] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

[9] P. Hanrahan and P. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In *SIGGRAPH 90 Conf. Proc.*, pages 215–223. Addison Wesley, 1990.

[10] K. Hormann and G. Greiner. MIPS: An efficient global parametrization method. In P.-J. Laurent, P. Sablonnière, and L. Schumaker, editors, *Curve and Surface Design: Saint-Malo 1999*, pages 153–162. Vanderbilt University Press, 2000.

[11] A. Hubeli and M. Gross. Multiresolution features extraction from unstructured meshes. In *Proc. of IEEE Visualization Conf.*, 2001.

[12] M. Hurdal, P. Bowers, K. Stephenson, D. Sumners, K. Rehms, K. Schaper, and D. Rottenberg. Quasi-conformally flat mapping the human cerebellum. In *Proc. of MICCAI'99*, volume 1679 of *Lecture Notes in Computer Science*, pages 279–286. Springer-Verlag, 1999.

[13] T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. In *Symp. on Interactive 3D Graphics*, pages 209–216. ACM, 2001.

[14] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH 96 Conf. Proc.*, pages 313–324. Addison Wesley, 1996.

[15] F. Lazarus and A. Verroust. Level set diagrams of polyhedral objects. In *Proc. of Solid Modeling and Applications*, pages 130–140. ACM Press, 1999.

[16] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 98 Conf. Proc.*, pages 95–104. Addison Wesley, 1998.

[17] B. Lévy. Constrained texture mapping for polygonal meshes. In *SIGGRAPH 01 Conf. Proc.*, pages 417–424. ACM Press, 2001.

[18] B. Lévy and J.-L. Mallet. Non-distorted texture mapping for sheared triangulated meshes. In *SIGGRAPH 98 Conf. Proc.*, pages 343–352. Addison Wesley, 1998.

[19] P. Lienhardt. Extension of the notion of map and subdivisions of a 3D space. In *Proc. of 5th Symp. on Theo. Aspects in Comp. Sci.*, pages 301–311, 1988.

[20] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *SIGGRAPH 93 Conf. Proc.*, pages 27–34. Addison Wesley, 1993.

[21] V. Milenkovic. Rotational polygon containment and minimum enclosure using only robust 2D constructions. *Computational Geometry*, 13(1):3–19, 1999.

[22] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle-packing-based module placement. In *Proc. of ICCAD*, pages 472–479. IEEE, 1995.

[23] H. Pedersen. Decorating implicit surfaces. In *SIGGRAPH 95 Conf. Proc.*, pages 291–300. Addison Wesley, 1995.

[24] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Math.*, 2(15), 1993.

[25] D. Piponi and G. Borshukov. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. In *SIGGRAPH 00 Conf. Proc.*, pages 471–478. ACM Press, 2000.

[26] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *SIGGRAPH 00 Conf. Proc.*, pages 465–470. ACM Press, 2000.

[27] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *SIGGRAPH 01 Conf. Proc.*, pages 409–416. ACM Press, 2001.

[28] A. Sheffer and E. de Sturler. Param. of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers*, 17(3):326–337, 2001.

[29] Y. Shinagawa, T. Kunii, and Y.-L. Kergosien. Surface coding based on Morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 1991.

[30] W. Tutte. Convex representation of graphs. In *Proc. London Math. Soc.*, volume 10, 1960.

[31] G. Zigelman, R. Kimmel, and N. Kiryati. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Transactions on Vis. and C.G.*, 2001.