Foundations and Trends® in
Computer Graphics and Vision
Vol. 2, No 2 (2006) 105–171

DOI: 10.1561/0600000011

now
the essence of knowledge

Mesh Parameterization Methods and Their Applications

Alla Sheffer[1], Emil Praun[2] and Kenneth Rose[3]

[1] *University of British Columbia, Canada, sheffa@cs.ubc.ca*
[2] *Google, USA*
[3] *University of British Columbia, Canada, kenrose@cs.ubc.ca*

Abstract

We present a survey of recent methods for creating piecewise linear mappings between triangulations in 3D and simpler domains such as planar regions, simplicial complexes, and spheres. We also discuss emerging tools such as global parameterization, inter-surface mapping, and parameterization with constraints. We start by describing the wide range of applications where parameterization tools have been used in recent years. We then briefly review the pertinent mathematical background and terminology, before proceeding to survey the existing parameterization techniques. Our survey summarizes the main ideas of each technique and discusses its main properties, comparing it to other methods available. Thus it aims to provide guidance to researchers and developers when assessing the suitability of different methods for various applications. This survey focuses on the practical aspects of the methods available, such as time complexity and robustness and shows multiple examples of parameterizations generated using different methods, allowing the reader to visually evaluate and compare the results.

1

Introduction

Given any two surfaces with similar topology, it is possible to compute a one-to-one and onto mapping between them. If one of these surfaces is represented by a triangular mesh, the problem of computing such a mapping is referred to as mesh parameterization [7, 35]. The surface that the mesh is mapped to is typically referred to as the parameter domain. Parameterizations between surface meshes and a variety of domains have numerous applications in computer graphics and geometry processing as described below. In recent years numerous methods for parameterizing meshes were developed, targeting diverse parameter domains and focusing on different parameterization properties. This survey reviews the various parameterization methods, summarizing the main ideas of each technique and focusing on the practical aspects of the methods. It also provides examples of the results generated by many of the more popular methods. When several methods address the same parameterization problem, the survey strives to provide an objective comparison between them based on criteria such as parameterization quality, efficiency, and robustness.

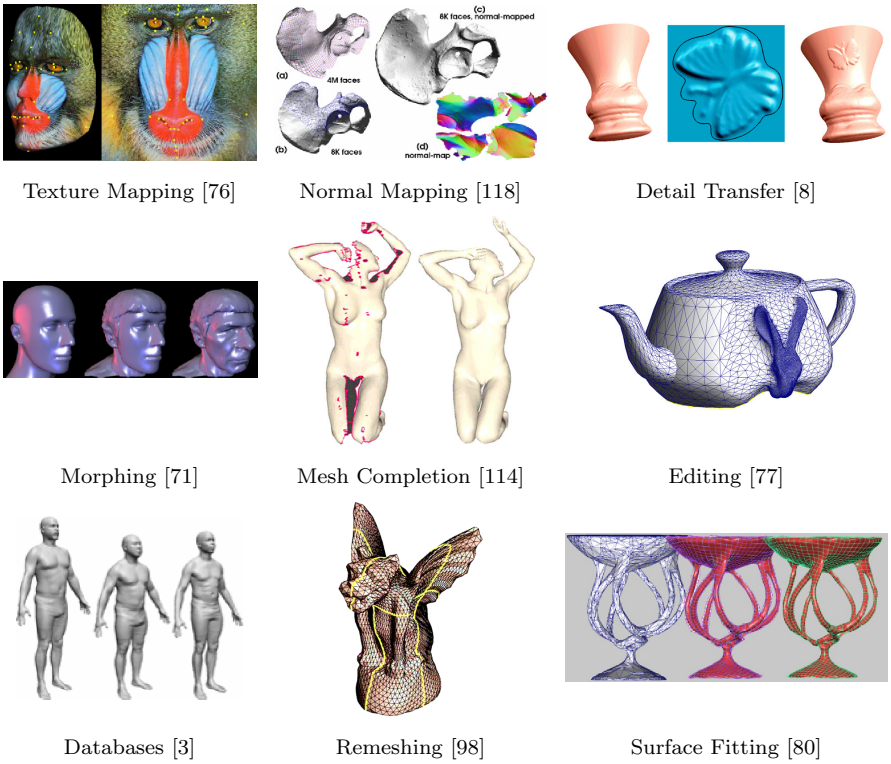
We start by surveying the applications which can benefit from parameterization in Section 1.1 and then in Section 2 briefly review

the terminology commonly used in the parameterization literature. The rest of the survey describes the different techniques available, classifying them based on the parameter domain used. Section 3 describes techniques for planar parameterization. Section 4 reviews methods for pre-processing meshes for planar parameterization by cutting them into one or more charts. Section 5 examines parameterization methods for alternative domains such as a sphere or a base mesh as well as methods for cross-parameterization between mesh surfaces. Section 6 discusses ways to introduce constraints into a parameterization. Finally, Section 7 summarizes the paper and discusses potential open problems in mesh parameterization.

1.1 Applications

Surface parameterization was introduced to computer graphics as a method for mapping textures onto surfaces [7,84]. Over the last decade, it has gradually become a ubiquitous tool, useful for many mesh processing applications, discussed below (Figure 1.1).

Detail Mapping Detailed objects can be efficiently represented by a coarse geometric shape (polygonal mesh or subdivision surface) with the details corresponding to each triangle stored in a separate 2D array. In traditional texture mapping, the detail is the local albedo of a Lambertian surface. Texture maps alone can enrich the appearance of a surface in a static picture, but since neighboring pixels will have similar shadowing, objects may still look flat in animations with varying lighting conditions. Bump mapping stores small deviations of the pointwise normal from that of the smooth underlying surface and uses the perturbed version during shading [13]. Normal mapping [118, 130] is a similar technique that replaces the normals directly rather than storing a perturbation. As the light direction changes, the shading variations produced by the normal perturbations simulate the shadows caused by small pits and dimples in the surface. Since the actual geometry of the object is not modified, the silhouettes still look polygonal or smooth. Displacement mapping addresses this problem by storing small local deformations of the surface, typically in the direction of the normal.



Texture Mapping [76] Normal Mapping [118] Detail Transfer [8]

Morphing [71] Mesh Completion [114] Editing [77]

Databases [3] Remeshing [98] Surface Fitting [80]

Fig. 1.1 Parameterization applications.

Recent techniques [75,93,96] model a thick region of space in the neighborhood of the surface by using a volumetric texture, rather than a 2D one. Such techniques are needed in order to model detail with complicated topology or detail that cannot be easily approximated locally by a height field, such as sparsely interwoven structures or animal fur. The natural way to map details to surfaces is using planar parameterization (Section 3).

Detail Synthesis While the goal of texture mapping is to *represent* the complicated appearance of 3D objects, several methods make use of mesh parameterization to *create* the local detail necessary for a rich appearance. Such techniques can use as input flat patches with sample detail [92, 97, 119, 127, 129, 131]; parametric or procedural models; or

direct user input and editing [17, 57]. The type of detail can be quite varied and the intermediate representations used to create it parallel the final representations used to store it.

Morphing and Detail Transfer A map between the surfaces of two objects allows the transfer of detail from one object to another [81, 99, 121], or the interpolation between the shape and appearance of several objects [2, 63, 66, 71, 109]. By varying the interpolation ratios over time, one can produce morphing animations. In spatially varying and frequency-varying morphs, the rate of change can be different for different parts of the objects, or different frequency bands (coarseness of the features being transformed) [63,66,71]. Such a map can either be computed directly or, as more commonly done, computed by mapping both object surfaces to a common domain (Sections 5 and 6).

In addition to transferring the static appearance of surfaces, inter-surface parameterizations allow the transfer of animation data between shapes, either by transferring the local surface influence from bones of an animation rig, or by directly transferring the local affine transformation of each triangle in the mesh [122].

Mesh Completion Meshes from range scans often contain holes and multiple components. Lévy [77] uses planar parameterization to obtain the natural shape for hole boundaries and to triangulate those. In many cases, prior knowledge about the overall shape of the scanned models exists. For instance, for human scans, templates of a generic human shape are readily available. Allen *et al.* [3], and Anguelov *et al.* [6] use this prior knowledge to facilitate completion of scans by computing a mapping between the scan and a template human model. Kraevoy and Sheffer [67] develop a more generic and robust template-based approach for completion of any type of scans. The techniques typically use an inter-surface parameterization between the template and the scan (Sections 5 and 6).

Mesh Editing Editing operations often benefit from a local parameterization between pairs of models. Biermann *et al.* [8] use local parameterization to facilitate cut-and-paste transfer of details between models.

They locally parameterize the regions of interest on the two models in 2D and overlap the two parameterizations. They use the parameterization to transfer shape properties from one model to the other. Sorkine *et al.* [121] and Lévy [77] use local parameterization for mesh composition in a similar manner. They compute an overlapping planar parameterization of the regions near the composition boundary on the input models and use it to extract and smoothly blend shape information from the two models.

Creation of Object Databases Once a large number of models are parameterized on a common domain (Sections 5 and 6), one can perform an analysis determining the common factors between objects and their distinguishing traits. For example on a database of human shapes [3], the distinguishing traits may be gender, height, and weight, while a database of human faces may add facial expressions [10–12, 85]. Objects can be compared against the database and scored against each of these dimensions, and the database can be used to create new plausible object instances by interpolation or extrapolation of existing ones.

Remeshing There are many possible triangulations that represent the same shape with similar levels of accuracy. Some triangulation may be more desirable than others for different applications. For example, for numerical simulations on surfaces, triangles with a good aspect ratio (that are not too small or too "skinny") are important for convergence and numerical accuracy. One common way to remesh surfaces, or to replace one triangulation by another, is to parameterize the surface, then map a desirable, well-understood, and easy to create triangulation of the domain back to the original surface. For example, Gu *et al.* [41] use a regular grid sampling of a planar square domain, while subdivision based methods [49, 63, 72] use regular subdivision (usually one-to-four triangle splits) on the faces of a simplicial domain. Such locally regular meshes can usually support the creation of smooth surfaces as the limit process of applying subdivision rules. To generate high quality triangulations Desbrun *et al.* [26] parameterize the input mesh in the plane and then use planar Delaunay triangulation to obtain

a high quality remeshing of the surface. One problem these methods face is the appearance of visible discontinuities along the cuts created to facilitate the parameterization.

Surazhsky and Gotsman [123] avoid global parameterization, and instead use local parameterization to move vertices along the mesh as part of an explicit remeshing scheme. Ray *et al.* [102] use global periodic parameterization to generate a predominantly quadrilateral mesh directly on the 3D surface. Dong *et al.* [26] use a parameterization induced by the Morse complex to generate a quad only mesh of the surface.

More details on the use of parameterization for remeshing can be found in a recent survey by Alliez *et al.* [5].

Mesh Compression Mesh compression is used to compactly store or transmit geometric models [4]. As with other data, compression rates are inversely proportional to the data entropy. Thus higher compression rates can be obtained when models are represented by meshes that are as regular as possible, both topologically and geometrically. Topological regularity refers to meshes where almost all vertices have the same degree. Geometric regularity implies that triangles are similar to each other in terms of shape and size, and vertices are close to the centroid of their neighbors. Such meshes can be obtained by parameterizing the original objects and then remeshing with regular sampling patterns [41, 52]. The quality of the parameterization directly impacts the compression efficiency.

Surface Fitting One of the earlier applications of mesh parameterization is surface fitting [32,51,54,80,82]. Many applications in geometry processing require a smooth analytical surface to be constructed from an input mesh. A parameterization of the mesh over a base domain significantly simplifies this task. Earlier methods either parameterized the entire mesh in the plane [32] or segmented it and parameterized each patch independently (Sections 3 and 4). More recent methods [51,80,82] focus on constructing smooth global parameterizations (Section 5.1) and use those for fitting, achieving global continuity of the constructed surfaces.

Modeling from Material Sheets While computer graphics focuses on virtual models, geometry processing has numerous real-world engineering applications. Particularly, planar mesh parameterization is an important tool when modeling 3D objects from sheets of material, ranging from garment modeling to metal forming or forging [7, 60, 86, 88]. All of these applications require the computation of planar patterns to form the desired 3D shapes. Typically, models are first segmented into nearly developable charts (Section 4), and these charts are then parameterized in the plane (Section 3).

Medical Visualization Complex geometric structures are often better visualized and analyzed by mapping the surface normal-map, color, and other properties to a simpler, canonical domain. One of the structures for which such mapping is particularly useful is the human brain [42,50,56]. Most methods for brain mapping use the fact that the brain has genus zero, and visualize it through spherical [42,50] (Section 5.2) or planar [56] (Section 3) parameterization.

Given the vast range of processing techniques that have benefited from parameterization, we expect that many more applications can utilize it as a powerful processing tool.

2

Terminology

Before proceeding to describe various parameterization techniques in the next section, we first briefly establish some terminology. We are concerned with the parameterization of triangle meshes. The topology of such meshes is typically represented as a simplicial complex: a set of 1-, 2-, and 3-element subsets of a set V of labels, corresponding respectively to the vertices, edges, and triangles of the mesh. The geometry of the mesh is represented as 3D coordinates associated with each of the vertices $c{:}V \rightarrow R^3$, making edges correspond to (open) line segments in 3D, and mesh triangles to (open) triangles in 3D.

The purpose of mesh parameterization is to obtain a map between such a mesh and a triangulation of a domain. The map is piecewise linear, associating each triangle of the original mesh with a triangle in the domain. An important goal of parameterization is to obtain *bijective* (invertible) maps, where each point on the domain corresponds to exactly one point of the mesh. Many applications of planar parameterization can, with some modifications, handle global domain overlaps (Figure 2.1(a)). For such applications the bijectivity requirement can be weakened, requiring only *local* rather than *global* bijectivity. Local bijectivity [118] requires a map of any sufficiently small region of the

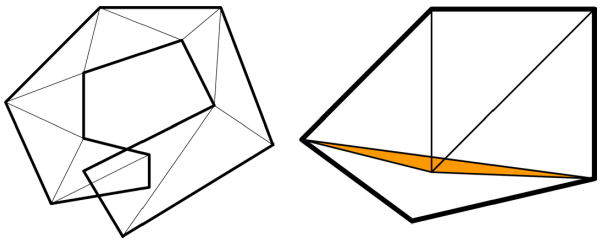


Fig. 2.1 Non-bijective parameterizations: (a) planar embedding with a global overlap; (b) planar embedding with a local overlap. The normal of the highlighted flipped triangle is inverted with respect to the other triangle normals.

mesh to be bijective. This condition is violated when the mappings of adjacent mesh triangles intersect, in this case the parameterization is said to "fold over" or contain "triangle flips" (Figure 2.1(b)).

The geometric shape of the domain triangles will typically be slightly different than the shape of the original triangles, resulting in angle and area distortion. Applications typically try to minimize the distortion for the whole mesh in a least squares sense. Very few meshes admit *isometric,* namely zero distortion, parameterizations. For example, only developable surfaces (such as cylindrical or conical sheets) admit planar isometric parameterizations. Maps that minimize the angular distortion, or *shear*, are called *conformal*, and maps that minimize area distortion are called *authalic*. Often conformal maps are also called *harmonic*, though, as shown by Floater and Hormann [35], the two are not equivalent. These terms are borrowed from differential geometry of smooth surfaces, where Riemann's theorem guarantees that conformal maps (with zero angular distortion) always exist, mapping any infinitesimally small circle on the surface to a circle on the domain, and thus preserving angles, but allowing the scale factor of the transformation to vary across the map. The Riemann theorem does not hold for meshes. For instance, when considering planar parameterization, the sum of angles around an interior mesh vertex in 3D can vary, while the sum of angles around a vertex in the plane is always 2π. Some conformal parameterization methods applied to a series of progressively denser meshes of the same object (with smaller and smaller

triangles, obtained through subdivision) will converge in the limit to a smooth conformal map. The conformality of a mesh can be measured in multiple different ways [25,54,79,113], resulting in different functionals to be optimized. For instance, Hormann and Greiner [54] consider the minimal and maximal eigenvalues γ and Γ of the first fundamental form of the mapping. Alternatively, Sheffer and de Sturler [113] directly measure the difference between the corresponding angles in the mesh and domain triangles.

As pointed out by Floater and Hormann [34], though authalic parameterizations are achievable, they are not very useful by themselves, as they allow extreme angular and linear distortion. Thus, methods that consider area preservation [24, 25] typically balance it with angle preservation.

Other metrics of parameterization distortion measure the preservation of distances across the mesh [107, 138, 140]. The *stretch* metrics proposed by Sander *et al.* [107] are now commonly used in the graphics community as standard measures of distance preservation. Sander *et al.* [107] observe that the linear map over each triangle can be decomposed into a translation, a rotation, and a non-uniform scale along two orthogonal axes. The two scale factors $0 \leq \gamma \leq \Gamma$ are the singular values of the transformation matrix, or the square roots of the eigenvalues of the integrated metric tensor (the transpose of the matrix times the matrix itself). Intuitively, the linear transformation map will stretch a unit circle to an ellipse with axes γ and Γ. The L_∞ stretch for a triangle is defined by Sander *et al.* [107] as $\max(\gamma,\Gamma) = \Gamma$, while the L_2 stretch is defined as $\sqrt{(\gamma^2 + \Gamma^2)/2}$. The name for the stretch metric comes from applications that map signals with regular sampling in the domain to 3D surfaces; these applications want to minimize the stretch of the signal over the surface, or the space between the locations of mapped samples. In other words, stretch penalizes undersampling the mesh, but not oversampling it.

When parameterizations are used to resample or compress 3D objects, the quality of the reconstruction can be measured by the symmetric Hausdorff metric between the original and reconstructed meshes. This metric measures the maximum distance between any point on either mesh and its projection on the other mesh. In practice most

people use an RMS-average of these point distances. For compression schemes that allow trade-offs between the bit rate (file size) and approximation accuracy, the performance is measured using rate-distortion curves, plotting one against the other.

Keeping these definitions in mind, we now proceed to survey the parameterization techniques available.

3

Parameterization of Topological Disks

The early papers to address parameterization for computer graphics applications were interested in planar parameterization of meshes with disk-like topology. The first application for such parameterizations was texture mapping. More recent applications include mapping of other surface properties such as normals or BRDFs, and mesh processing operations such as remeshing and compression. Two recent surveys [34, 35] list more than 20 different planar parameterization techniques. Both surveys focus on the mathematical aspects of these techniques. To avoid unnecessary overlaps, we will address the more practical considerations, such as the suitability of the techniques for computer graphics applications in terms of distortion (type and amount), robustness, and efficiency. In our discussion, we classify the methods based on the type of parametric distortion minimized.

Planar parameterization of 3D surfaces inevitably creates distortion in all but special cases. A well-known result from differential geometry is that for a general surface patch there is no distance-preserving (isometric) parameterization in the plane [26]. Distance-preserving parameterizations exist only for developable surfaces: that is, surfaces with

zero Gaussian curvature. Cutting the surface into charts or introducing seams (as discussed in Section 4), can reduce this distortion.

We classify planar parameterizations into roughly four groups: methods that ignore distortion altogether (Section 3.1), methods that minimize angular distortion (Section 3.2), methods that minimize stretch (Section 3.3), and methods that minimize area distortion (Section 3.4). There are also several techniques providing tools for achieving a trade-off between different types of distortion (Section 3.5).

Ideally, most parameterization applications work best on zero distortion parameterizations, though most are tolerant to some amount of distortion, some being more tolerant to shear and others to stretch. In general, applications that depend on regular grids for sampling, such as different types of detail mapping and synthesis, as well as compression and regular resampling schemes (e.g., geometry images [41]), tend to perform better on stretch minimizing parameterizations, since stretch is directly related to under-sampling. In contrast, applications based on irregular sampling, such as remeshing [25], are very sensitive to shearing, but can handle quite significant stretch. When acceptable levels of shear or stretch are not attainable because a surface is too complex, the surface needs to be cut prior to parameterization (Section 4) in order to achieve acceptable distortion.

In addition to distortion, several other factors should be considered when choosing a parameterization method for an application at hand:

- **Free versus fixed boundary** Many methods assume the boundary of the planar domain is pre-defined and convex. Fixed-boundary methods typically use very simple formulations and are very fast. Such methods are well suited for some applications, for instance those that utilize a base mesh parameterization, see Section 5.1. Free-boundary techniques, which determine the boundary as part of the solution, are often slower, but typically introduce significantly less distortion.
- **Robustness** Most applications of parameterization require it to be bijective. For some applications local bijectivity (no triangle flips) is sufficient, while others require global

Table 3.1 Planar method summary.

Method	Distortion minimized	Boundary	Bijectivity	Complexity
Uniform [128]	None	Fixed, convex	Yes	Linear
Harmonic [28]	Angles	Fixed, convex	No	Linear
Shape preserving [32]	Angles	Fixed, convex	Yes	Linear
Mean-value [33]	Angles	Fixed, convex	Yes	Linear
LSCM/DCP [25, 79]	Angles (&Area)	Free	No	Linear
ABF/ABF++ [113, 118]	Angles	Free	Local (no flips)	Nonlinear
MIPS [54]	Angles	Free	Yes	Nonlinear
Circle patterns [62]	Angles	Free	Local (no flips)	Nonlinear (unique minimum)
Stretch minimizing [107]	Distances	Free	Yes	Nonlinear
MDS [140]	Distances	Free	No	Nonlinear
Degener *et al.* [24]	Areas	Free	Yes	Nonlinear

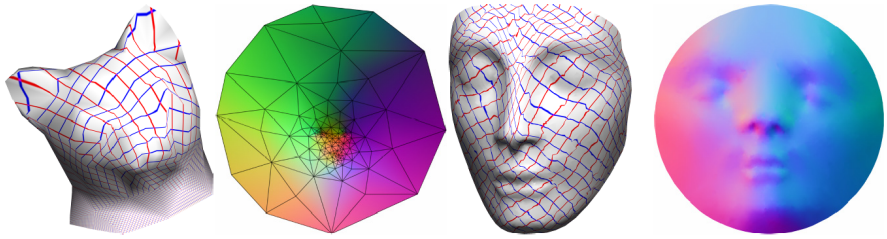
bijectivity conditions (the boundary does not self-intersect). Only a subset of the parameterization methods can guarantee local or global bijectivity. Some of the others can guarantee bijectivity if the input meshes satisfy specific conditions.

- **Numerical Complexity** The existing methods can be roughly classified according to the optimization mechanism they use into linear and nonlinear methods. Linear methods are typically significantly faster and simpler to implement. However, as expected the simplicity usually comes at the cost of increased distortion.

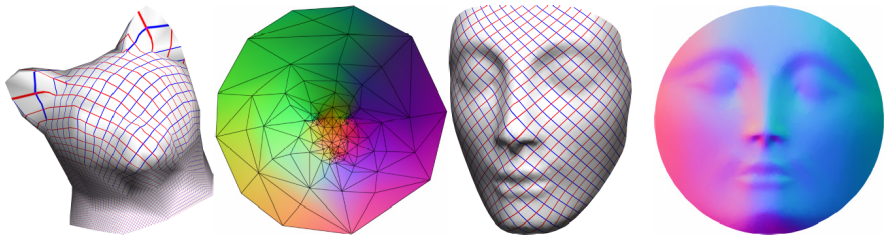
Table 3.1 provides a summary of a number of popular recent methods with respect to these four criteria. Figures 3.1–3.4 and Table 3.2 provide distortion and runtime comparison between some of the more popular recent methods.

3.1 Planar Mesh Embedding

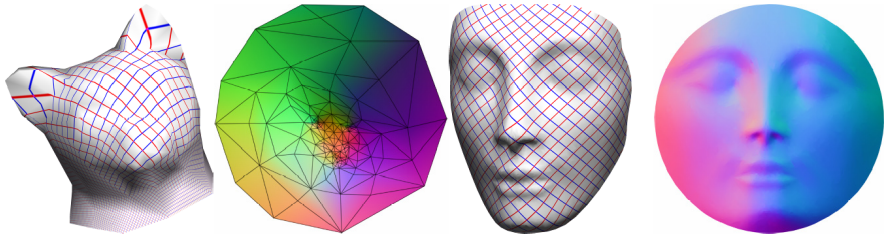
One of the oldest methods referenced in the context of mesh parameterization is the graph embedding method of Tutte [128]. Tutte's formulation of graph embedding directly applies to triangular meshes. It also



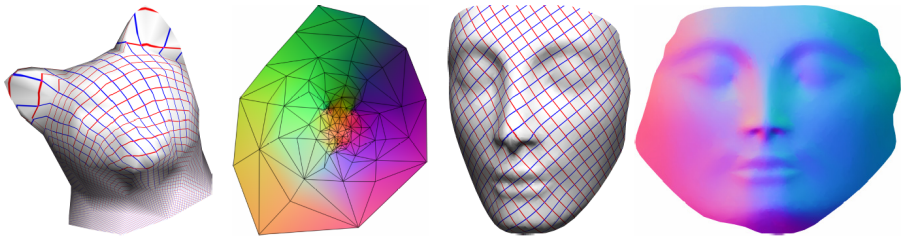
Parameterization with uniform weights [128] on a circular domain.



Parameterization with harmonic weights [28] on a circular domain.

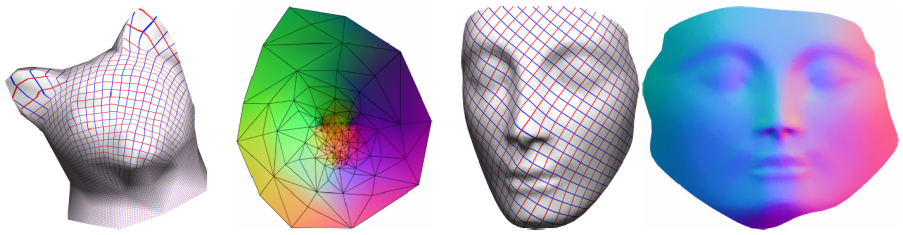


Parameterization with mean value weights [33] on a circular domain.

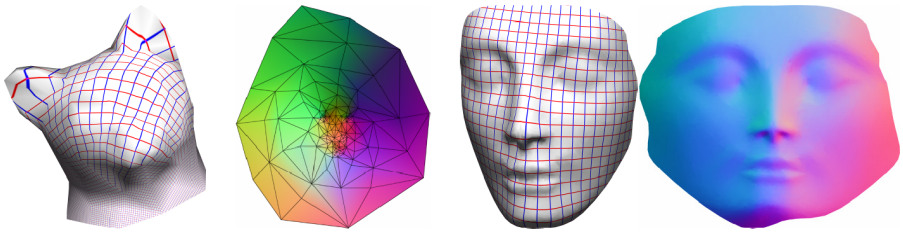


Parameterization with LSCM [79].

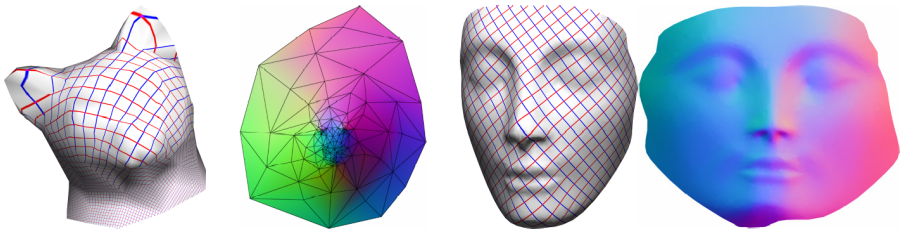
Fig. 3.1 Linear parameterization methods.



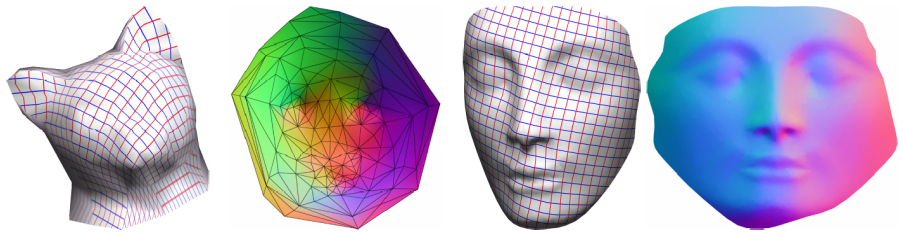
Parameterization with MIPS [54].



Parameterization with ABF++ [118].



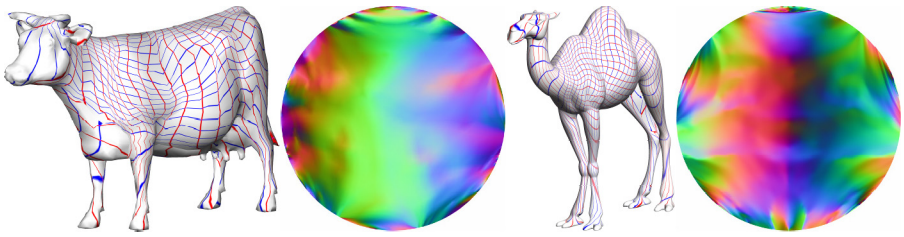
Parameterization with circle patterns [62].



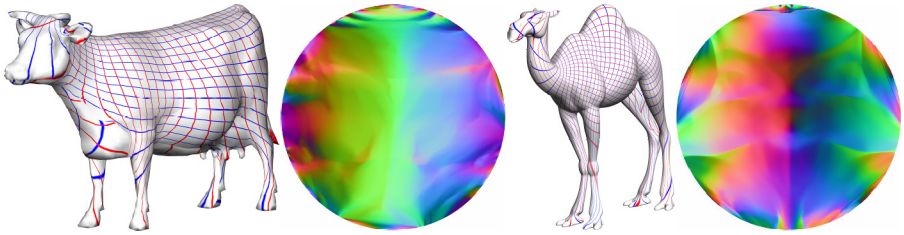
Stretch minimizing parameterization [107].

Fig. 3.2 Nonlinear parameterization methods.

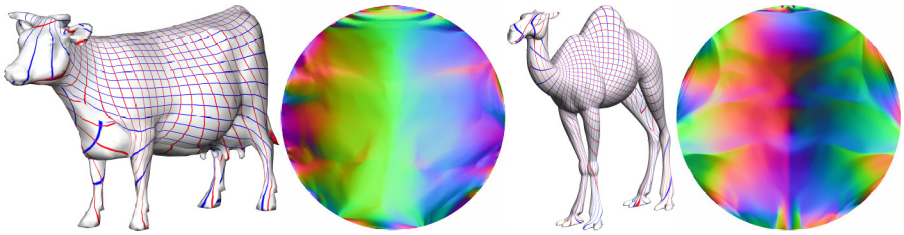
provides a general framework employed by many more-recent methods. The graph embedding formulation uses a two stage procedure. First, the boundary vertices of the mesh are mapped to the boundary of a convex region in 2D. Then the positions of the rest of the vertices are



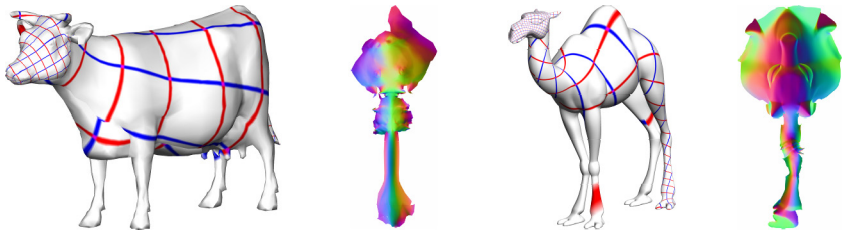
Parameterization with uniform weights [128] on a circular domain.



Parameterization with harmonic weights [28] on a circular domain.



Parameterization with mean value weights [33] on a circular domain.

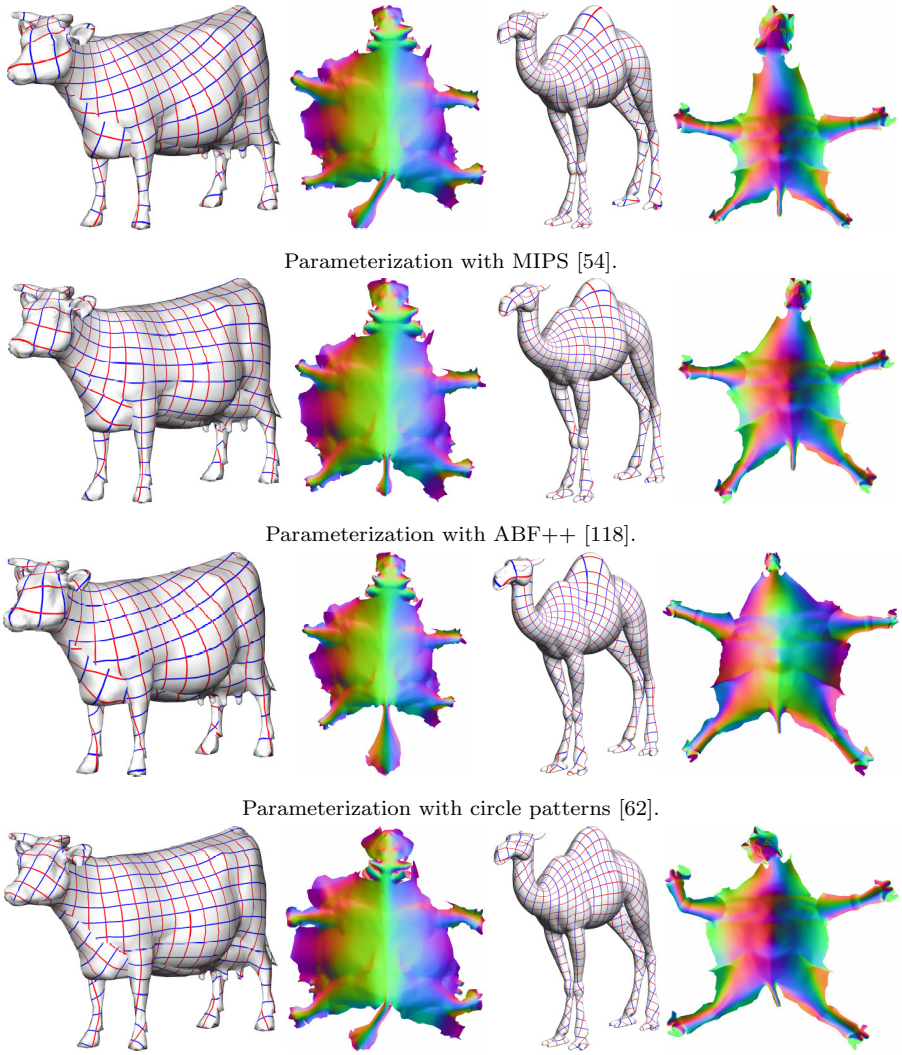


Parameterization with LSCM [79].

Fig. 3.3 Linear parameterization methods on closed objects with a seam.

obtained by solving a linear system of the form

$$\begin{aligned} &Lu = 0, \quad Lv = 0 \\ &L_{i,j} = \begin{cases} -\sum_{k \neq i} L_{i,k} & i = j \\ w_{ij} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} . \end{aligned} \tag{3.1}$$



Parameterization with MIPS [54].

Parameterization with ABF++ [118].

Parameterization with circle patterns [62].

Stretch minimizing parameterization [107].

Fig. 3.4 Nonlinear parameterization for closed objects with a seam.

The system is solved independently for the u and v coordinates. The parameter values of the fixed boundary vertices are used to specify the boundary conditions for the system. The weights w_{ij} are defined for each edge of the mesh. It was proven [32, 128], that if the weights are positive and the matrix is symmetric, then the obtained

parameterization is guaranteed to be bijective. Tutte [128] used uniform unit weights, setting $w_{ij} = 1$ iff (i, j) is an edge in the mesh. Note that this setting defines L as the classical graph Laplacian matrix [19]. While the resulting parameterization is provably bijective, it does not preserve any shape properties of the mesh (Figures 3.1 and 3.3, top row).

Another well known, but less frequently used, mesh embedding technique is based on the circle packing theorem [104]. Given a triangular mesh, a circle packing is a collection of circles, one for each mesh vertex, such that two circles are tangential if there is a mesh edge between their associated vertices. By connecting the centers of these circles we obtain a planar embedding. The circle packing theorem states that for any given triangular mesh with disk topology and for any selection of radii r associated with the boundary vertices of the mesh, there exists a unique (up to symmetries) circle packing of the mesh with boundary circles having these radii. Collins and Stephenson [23] provide a constructive technique to use the theorem to obtain mesh embeddings based only on combinatorial mesh information. Thus the method is guaranteed to generate bijective embeddings but does not preserve shape properties.

3.2 Angle-Preserving Parameterization

Many of the disk-parameterization methods are focused on minimizing the angular distortion, or shear, of the parameterization. Several applications require angle-preserving parameterization; for instance, remeshing replaces one triangulation with another of better quality, typically by mapping a regular triangulation of the domain onto the mesh. The quality is often defined in terms of angles: numerical simulations are adversely affected by small angles, while good compression depends on regular structures with triangles similar to their neighbors.

Angle-preserving parameterizations are efficient to obtain and are also often suitable for other applications, such as texture mapping, as long as the stretch of the parameterization is relatively low. This is often the case when the parameterized surfaces are not too far from developable. Riemann's theorem [26] states that for smooth surfaces a conformal planar parameterization exists for any planar domain. Thus,

since meshes are often viewed as approximations of smooth surfaces, we can argue that it is possible to map them to the plane with very little angular distortion.

Several parameterization techniques [28,32,33] use an approach similar to Tutte's [128]. They first map the boundary of the surface to the boundary of a convex domain in 2D and then obtain the parameterization for the rest of the vertices solving a system of equations (Equation (3.1)), independently for the u and v coordinates. The main difference between these methods is in the values w_{ij} assigned as edge weights. The choice of weights influences both the distortion and the bijectivity of the parameterization.

The harmonic or cotangent weights [28, 94] are perhaps the most widely known in the graphics community. They define the matrix elements as:

$$w_{ij} = (\cot \alpha_{ij} + \cot \beta_{ij})/2, \tag{3.2}$$

where α_{ij} and β_{ij} are the opposite angles in the two triangles shared by the edge (i, j) as shown in Figure 3.5. These weights are derived from a finite-element description of harmonic energy and therefore aim at reducing the angular distortion of the parameterization. The greatest drawback of harmonic parameterization is that if the mesh contains obtuse angles, the weights can be negative, and, as a result, the parameterization can be non-bijective. Hence this formulation is not suitable, as is, for applications where bijectivity is a major concern. In

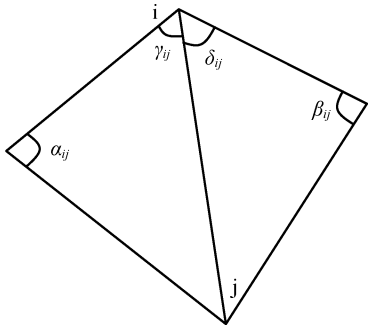


Fig. 3.5 Angles used for harmonic and mean-value weights.

practice many methods suggest first improving the mesh quality by bisecting obtuse angles or flipping edges. It has been proven that if the mesh satisfies the Delaunay criterion, even if it contains obtuse triangles, the parameterization obtained using the cotangent weights (Equation (3.2)) will always be bijective. Recently, Kharevych *et al.* [62] suggested using an "intrinsic" Delaunay triangulation of the surface as an input to the harmonic mapping to guarantee bijectivity. Figures 3.1 and 3.3 show a few examples of parameterizations generated using harmonic parameterization on a circular domain.

The shape-preserving parameterization [32] is also based on discrete harmonic mapping. It uses weights which are positive and symmetric thus ensuring that the parameterization is always bijective. However, the weights vary non-smoothly, i.e., they have derivative discontinuities, as the vertex i moves inside its one-ring polygon. Guskov [47] modifies the shape preserving parameterization [32] by introducing an extra term that allows an anisotropic parameterization stretched in the direction of a given surface direction field.

The mean-value weights also proposed by Floater [33] are much simpler and vary smoothly

$$w_{ij} = (\tan(\gamma_{ij}/2) + \tan(\delta_{ij}/2))/\left\|c(i) - c(j)\right\|,$$

where $c(i)$ and $c(j)$ are the 3D positions of vertices i and j, and γ_{ij} and δ_{ij} are the angles in the two triangles shared by the edge (i, j) shown in Figure 3.5. The mean-value weights are always positive. The resulting matrix is not symmetric ($w_{ij} \neq w_{ji}$) thus the bijectivity proof of Tutte [128] does not directly apply to this formulation. Nevertheless, Floater [33] proves that mean-value parameterization is guaranteed to be bijective. As shown in Figures 3.1 and 3.3 (third row) the mean-value weights lead to parameterizations with small angular distortion. However, it was demonstrated [135] that they can in some cases introduce larger angular distortion than the more commonly used harmonic weights [28].

All of these techniques are very efficient and simple to implement as they only require solving a single linear system. However, since the parameterization distortion depends on how close the actual boundary shape matches the 2D domain shape, the fixed-boundary techniques

perform poorly when the 3D meshes have non-convex boundaries (Figure 3.3), or boundaries that differ significantly from the specified boundary of the planar domain. Thus, these techniques work best when the inputs have well-shaped nearly convex boundaries. For instance, in a base-mesh setting (Section 5.1) the input mesh is first segmented into nearly triangular patches. In this setting, the fixed-boundary methods introduce very little distortion when mapping the patches to the corresponding triangular base mesh faces.

To reduce the distortion, the boundary of the 2D domain can be computed as part of the solution. Lee *et al.* [74] free the domain boundary by introducing one or more layers of triangles around the original boundary in 3D, creating a *virtual* boundary (Figure 3.6(a)). The virtual boundary is fixed onto a given convex polygon and the mapping for the rest of the vertices is computed using Floater's method [32]. Since the original boundary vertices are free to move, the method obtains a parameterization with less distortion than with the original convex combination approach (Figure 3.6). A similar idea was used by Kos and Varady [65]. Zhang *et al.* [137] use *scaffold triangles* in the virtual boundary that only contribute a flip penalty to a stretch-like optimization metric.

Recently, two explicit formulations of free-boundary, linear parameterization, LSCM and DCP, were independently proposed by

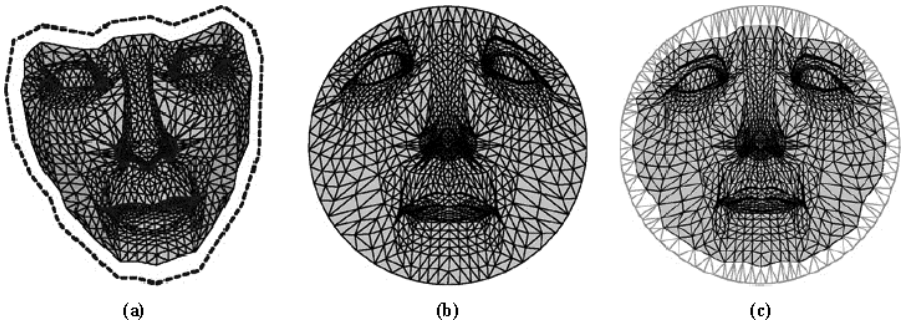


Fig. 3.6 (a) Adding a virtual boundary to the original mesh. (b) Shape Preserving [32] parameterization of the original mesh. (c) Parameterization of the original mesh and its virtual boundary [74]. The virtual boundary vertices are fixed, allowing the real boundary vertices to move.

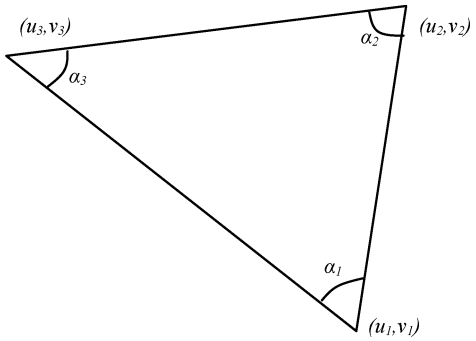


Fig. 3.7 LSCM [79] notations.

Lévy *et al.* [79] and Desbrun *et al.* [25]. Using different formulations of harmonic energy, both sets of authors derived equivalent formulations for free-boundary parameterization that aim to minimize angular distortion. The formulation of Lévy *et al.* [79] is based on the observation that given the angles α_1, α_2, α_3 of a planar triangle (Figure 3.7) the following holds:

$$(u_3, v_3) - (u_1, v_1) = \frac{\sin\alpha_2}{\sin\alpha_3} R^{\alpha_1}[(u_2, v_2) - (u_1, v_1)], \tag{3.3}$$

where u_i, v_i are the planar coordinates of the triangle vertices and R^α a rotation matrix with angle α. To minimize angular distortion the authors plug the original 3D angles into the formula and minimize the sum of square distances between the left and right sides of Equation (3.3). They fix two vertices to avoid the degenerate solution where all the vertices are at one point. Interestingly, the conditions specified by these two methods for interior mesh vertices are equivalent to the discrete harmonic energy formulation (Equation (3.2)). Thus the difference in the results can be viewed as a difference in the imposed boundary conditions.

Figures 3.1 and 3.3 (bottom row) show some models parameterized with LSCM. As demonstrated LSCM/DCP introduce significantly less distortion than fixed boundary approaches, particularly near the domain boundaries. The methods do not guarantee local or global bijectivity, and can theoretically result in both flipped triangles and global

overlaps. The HLSCM method [101] describes a mechanism for speeding up the solution process for LSCM using a hierarchical solver.

On meshes with high curvature, nonlinear conformal methods, discussed below, introduce significantly less stretch than DCP/LSCM (Figure 3.4). The MIPS method [54, 55] optimizes a nonlinear functional that measures mesh conformality (Figures 3.2 and 3.4, top row). To obtain a solution it starts with a harmonic fixed-boundary parameterization [32] as an initial guess. It then proceeds to move vertices one at a time to reduce the distortion metric. To prevent flips, vertices are only permitted to move inside the kernel of neighboring vertices. When moving boundary vertices, the method also checks for boundary overlaps and prevents those. Thus, it guarantees that the resulting parameterization remains globally bijective throughout the procedure. The authors advocate the use of conformal mapping for remeshing [69] and surface fitting [53].

Instead of defining a planar parameterization in terms of vertex coordinates, the ABF method [113,114] defines it in terms of the angles of the planar triangles. Sheffer and de Sturler [113] specify a set of constraints that angle values have to satisfy to define a planar triangular mesh. They search for angles that are as close as possible to the original 3D mesh angles and that satisfy those constraints. They then convert the solution angles into actual (u, v) vertex coordinates. The resulting parameterizations are guaranteed to have no flipped triangles (local bijectivity) but can contain global overlaps. The authors provided a mechanism for resolving such overlaps, but it has no guarantees of convergence. The original ABF method is relatively slow and suffers from stability problems in the angle-to-uv conversion stage for large meshes. ABF was augmented to yield ABF++ [118], a technique addressing both problems (Figures 3.2 and 3.4, second row). ABF++ introduces a stable angle-to-uv conversion and drastically speeds up the solution by introducing both direct and hierarchical solution approaches.

Zayer *et al.* [133] proposed to extend ABF by using additional constraints on the angles enforcing the parameter domain to have convex boundaries, thus guaranteeing global bijectivity. Zayer *et al.* [135] introduce an iterative free-boundary conformal method. They start with a fixed-boundary parameterization and then relax it. In their

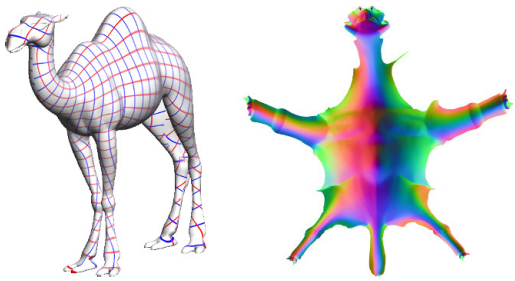


Fig. 3.8 Setting the boundary free [135].

comparisons, the method seems to be faster but introduce more distortion than ABF and MIPS (Figure 3.8).

Hurdal *et al.* [56] and Bowers and Hurdal [15] compute angle-preserving parameterizations using a variation of the circle-packing formulation described later by Collins and Stephenson [23]. They observe that, in the circle-packing setting, angle preservation corresponds to preservation of specified distances between circles. They construct patterns of non-intersecting circles, where each circle corresponds to a vertex of a triangulation. To preserve angles, they optimize distances between circles corresponding to neighboring vertices. The authors use the conformal mapping technique for medical visualization, focusing on visualizing the structure of the human brain.

Kharevych *et al.* [62] use a circle patterns approach where each circle corresponds to a mesh face. In contrast to classical circle packing, they use intersecting circles, with prescribed intersection angles. Given these angles, the circle radii follow as the unique minimizer of a convex energy. The method first computes the intersection angles using nonlinear constrained optimization and then finds the energy minimizer. Since the solution for the intersection angles is conformal only for a Delaunay triangulation, the authors employ a pre-processing stage that involves "intrinsic" Delaunay triangulation. At the final stage of the method, the computed angles are converted to actual uv-coordinates. The method supports equality and inequality constraints on the angles along the boundary of the planar parameter domain. Similar to ABF, the parameterization is locally bijective, but can contain global overlaps. As shown in Figures 3.2 and 3.4, the amount of

Table 3.2 Timings (s) of various parameterization techniques.

Method	Cat head (257 Δ)	Nefertiti (8071 Δ)	Cow (5804 Δ)	Camel (78K Δ)
Uniform [122]*	0.02	0.23	0.14	2.91
Harmonic [28]*	0.02	0.26	0.17	3.21
Mean-value [33]*	0.02	0.25	0.16	3.19
LSCM [79]*	0.03	0.38	0.20	5.28
ABF++ [118]*	0.06	1.87	0.77	36.31
MIPS [54]‡	1	8	5	83
Circle patterns [62]*	0.1	3.6	2.1	76.7
Stretch minimizing [107]	5.17†	12	8.5	127

†1.2 GHz Pentium M; * 3.0 GHz Pentium 4; ‡ 3.2 GHz Pentium 4.

distortion introduced by the method is comparable with that of other nonlinear conformal techniques. Kharevych *et al.* propose an extension of the method to global parameterization of meshes by introducing *cone singularites*, as described in Section 6.

Karni *et al.* [61] propose an interesting iterative method for resolving triangle flips in existing planar triangulations. The input to their method is a planar triangulation generated by one of the methods above. If the triangulation contains flipped triangles they employ an iterative vertex relocation procedure to eliminate the flips. Their method can also be used to support soft positional constraints in planar triangulations. The method reduces the number of flipped triangles does not guarantee that the final triangulation will be flip-free.

Table 3.2 lists the runtimes for the different conformal methods. We used the Graphite 3D modeling system [39] to time the fixed-boundary methods, LSCM and ABF++. For the other methods the timings were provided by the authors. As expected, linear techniques are about one order of magnitude faster than the nonlinear ones. Nevertheless, even the nonlinear methods are fairly fast taking less than two minutes to process average size models.

3.3 Distance Preserving Parameterization

In contrast to conformal parameterization, which always exists for smooth surfaces, stretch- or distance-preserving parameterization exists only for developable surfaces. Hence existing methods aim at

minimizing linear distortion rather than completely eliminating it. Early parameterization techniques [7, 78, 84] introduced distance minimizing formulations which were numerically quite complex and thus hard to minimize.

Sander *et al.* [107] introduce two metrics of parameterization stretch (L_2 and L_∞), described in Section 2. These two stretch metrics are now commonly used for comparing linear distortion between parameterizations [62, 118, 138]. They then proceed to minimize the L_2 metric using a solution mechanism similar to that of Hormann and Greiner [54], starting with a shape-preserving parameterization [32] and moving one vertex at a time (Figures 3.2 and 3.4, bottom row). To speed up the parameterization, the authors use a hierarchical top-down parameterization approach. Similar to MIPS, the stretch minimization method starts with non-self-intersecting parameter domains, therefore, it can effectively prevent both flips and global overlaps, thus guaranteeing globally bijective parameterization. Since the method's goal is to minimize an L_2 norm, it sometimes compensates for shrinkage in one direction by stretching in another, introducing shearing (see camel's neck in Figure 3.4, bottom row). Example runtimes for the method are listed in Table 3.2. The runtimes are slower but within the same order of magnitude as those of the nonlinear conformal methods. Sander *et al.* [106] introduced an extension of the method to signal specialized parameterization, where the user can influence the distribution of the distortion across the surface. This method assumes that the reconstructed signal is piecewise constant. Tewari *et al.* [125] assume that the reconstruction is piecewise linear, which provides greater sensitivity to signal detail.

Zigelman *et al.* [140] introduce a mesh parameterization method that aims to preserve the distances between all pairs of vertices on the surface. They first compute those distance using geodesics, and then use multi-dimensional scaling (MDS) to embed the mesh in the plane. The method performs quite well for surfaces that are close to developable. However, for complex surfaces, it often tends to fold the surface in the plane generating a non-bijective parameterization, since the folded solution provides better distance preservation [138].

Zhou *et al.* [138] also use an MDS or *isomap* based approach, introducing several mechanisms to speedup the computation process. If the

parameterized planar mesh contains folds, they segment the surface, and parameterize each chart independently (Section 4). Figure 3.9(b) shows the charts and the parameterization produced by the method.

In some settings it is possible to avoid solving a linear system and simply compute the parameterization one triangle at a time [86, 120]. However, for most meshes this technique must be combined with simultaneous generation of cuts in the mesh surface. We therefore choose to discuss it together with other cutting techniques in Section 4.

3.4 Area-Preserving Parameterization

As pointed out by Floater and Hormann [35], from a theoretical point of view, there are two independent qualities that can be minimized in a mapping: angular distortion and area distortion. For continuous surfaces, whereas conformal mappings are nearly unique, area-preserving mappings are not. Thus methods that search for *authalic*, namely area-preserving, mesh parameterizations typically introduce additional optimization terms or constraints on the parameterization. Desbrun *et al.* [25] use a similar derivation of their conformal DCP embedding to obtain a linear formulation for local area preservation. Degener *et al.* [24] develop a nonlinear formulation of per-triangle area preservation. Both papers then proceed to combine the area optimization metric with a conformality metric to allow for parameterizations that mediate between the two as explained in the next section.

3.5 Trade-Off Between Metrics

Many applications of parameterizations fail when either the shear or the stretch are extreme. However, they can tolerate a small amount of either type of distortion. Thus it is sometimes advisable to provide a parameterization that provides an adequate degree of trade-off between angle preservation and area, or length preservation.

Desbrun *et al.* [25] provide a linear formulation which supports a trade-off between angular and area distortion. It is not clear if the combined formulation can be used in a free-boundary setting. Degener *et al.* [24] extended the MIPS method [54] to find parameterizations

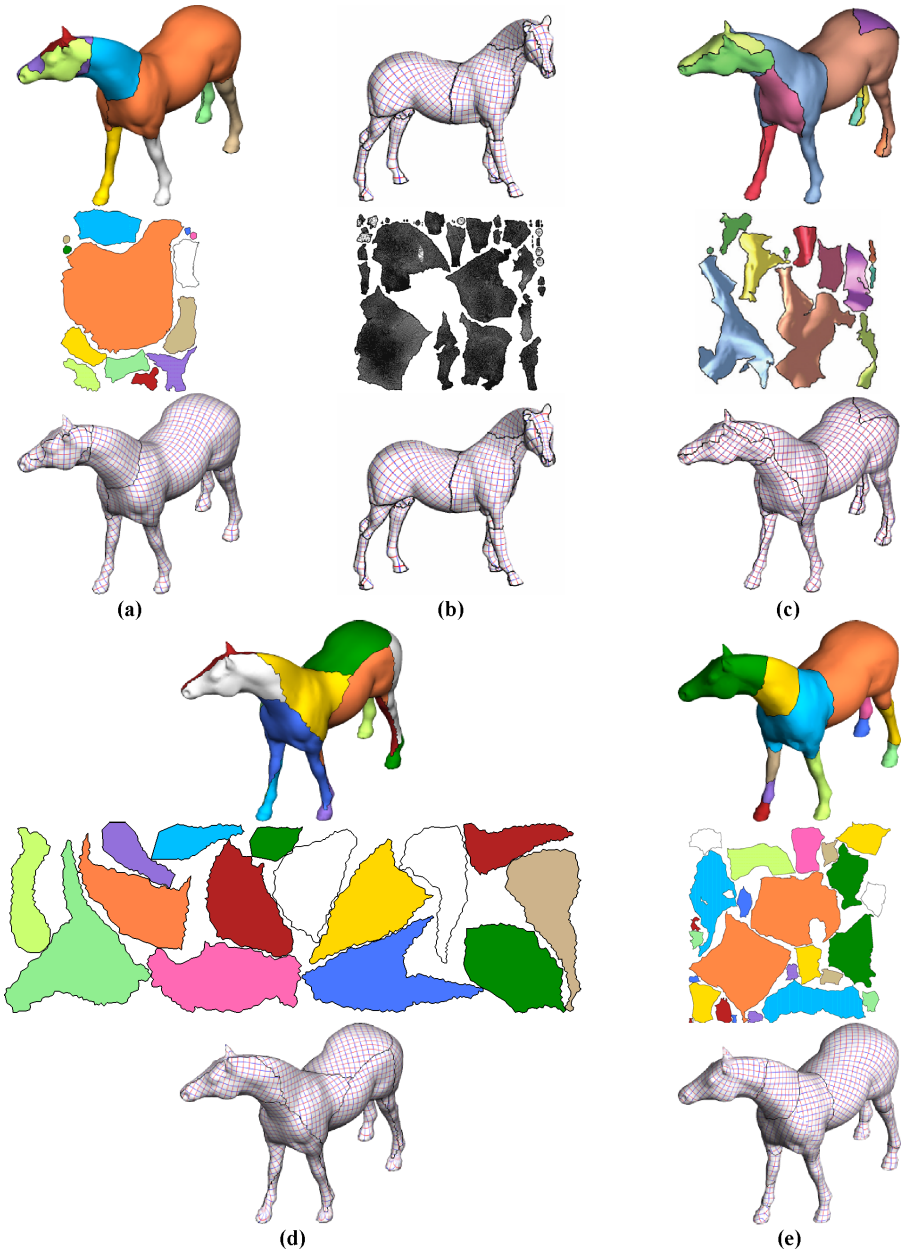


Fig. 3.9 Segmentation and texture atlases for (a) Isocharts [138] – 13 charts; (b) Lévy *et al.* [79] – 44 charts; (c) D-charts [60] – 12 charts; (d) multi-chart geometry images [108] – 15 charts; and (e) Zhang *et al.* [137] – 24 charts.

that mediate between angle and area deformation by modifying the minimized functional. They define the minimized energy to be a sum of per-triangle terms. Each term is a product of the MIPS harmonic energy per-triangle and an energy functional that measures area distortion. The powers of the components in the product provide the mediation between the two measures. The method maintains the bijectivity guarantees of the original MIPS algorithm. Clarenz *et al.* [20] provide a different set of nonlinear energy metrics allowing for trade-off between angle, area, and distance distortion measures.

In recent years, a number of researchers suggested using a two stage approach. First, a mesh is parameterized using one of the existing techniques listed above, and then a second, post-processing stage is applied to improve the parameterization. The improvement typically focuses on reducing some measure of distortion, or providing a trade-off between several distortion measures. The methods measure the distortion of parameterization produced by the first stage of the algorithm, and then adopt the parameters of the second stage to reduce this distortion.

Sheffer and de Sturler [115] first computed an angle-preserving parameterization using ABF [113]; they then used an overlay grid to compute the stretch of the resulting parameterization; finally, they smoothed the overlay grid to mitigate the stretch.

Yoshizawa *et al.* [132] and Zayer *et al.* [134] also start with a parameterization that minimizes angular distortion. They first use linear, fixed-boundary methods to compute one. They then compute the stretch of the parameterization; scale the weights used in the linear method by a mitigating factor and solve the linear system again. To scale the weights, Yoshizawa *et al.* [132] used the stretch metric of Sander *et al.* [107], while Zayer *et al.* [134] use a simpler construction based on inverse edge lengths.

4

Cutting/Chart Generation

Planar parameterization is only applicable to surfaces with disk topology. Hence, closed surfaces and surfaces with genus greater than zero have to be cut prior to planar parameterization. As previously noted, greater surface complexity usually increases parameterization distortion, independent of the parameterization technique used. To allow parameterizations with low distortion, the surfaces must be cut to reduce the complexity. Since cuts introduce discontinuities into the parameterization, a delicate balance between the conflicting goals of small distortion and short cuts has to be achieved. It is possible to use constrained parameterization techniques to reduce cross-cut discontinuities [68,139] as discussed in Section 6. Cutting and chart generation are most commonly used when computing parameterizations for mapping of textures and other signals onto the surface. They are also used for applications such as compression [41] and remeshing.

The techniques for cutting surfaces can be roughly divided into two categories: segmentation techniques which partition the surface into multiple charts (Section 4.1), and seam generation techniques which introduce cuts into the surface but keep it as a single chart (Section 4.2). Multiple charts created by segmentation typically have longer boundaries

than those created by seam cutting. However, they can often be more efficiently packed (Section 4.1.1) into a compact planar domain.

4.1 Mesh Segmentation

Depending on the application, mesh segmentation techniques use different criteria for creating charts [110]. Segmentation techniques, used for parameterization, break the surface into several charts such that the parametric distortion when parameterizing each chart is sufficiently low, while the number of charts remains small and their boundaries are kept as short as possible. Since planes are developable by definition, one possible approach is to segment the surface into nearly planar charts [21, 36, 84, 107, 108].

Earlier approaches are based on incremental clustering of mesh faces into charts. They typically start by selecting several initial *seed* mesh faces, and then grow the charts around those seeds, adding one mesh face at a time, until all the faces are classified as belonging to one chart or the other.

Recently, Sander *et al.* [108] introduced a segmentation method inspired by Lloyd quantization (Figure 3.9(d)). Their algorithm iterates between chart growing and reseeding stages. After each chart growing iteration, the method selects the best seeds with respect to each chart and repeats the process. The authors demonstrate that by iterating, they obtain better results than single pass methods such as that of Sander *et al.* [107].

Planes are a special type of developable surfaces. Thus for parameterization purposes planar segmentation is over-restrictive and usually generates more charts than necessary. Several recent approaches focused on developable segmentation instead [60, 79, 138]. Lévy *et al.* [79] proposed to detect high mean-curvature regions on the mesh and then generate charts starting from seeds which are farthest from those regions (Figure 3.9(b)). This approach tends to capture many developable regions, but can also introduce charts which are far from developable.

Zhou *et al.* [138] propose a segmentation method based on spectral analysis of the surface (Figure 3.9(a)). They compute a matrix of

geodesic mesh distances and then perform face clustering by growing charts around the n farthest points in a space defined by the dominant eigenvectors of the matrix.

Zhang *et al.* [137] use a Reeb graph connecting the critical points of the average geodesic distance to cut the mesh, reducing its genus to zero. The surface is further segmented into features by finding field iso-contours where the feature area increases significantly; then the features are classified as linear ellipsoids, flat ellipsoids, or spheres, and cut using a lengthwise, circular, or, respectively, a "baseball seam" cut. The resulting pieces are close to being developable and can be parameterized with little distortion (Figure 3.9(e)).

Similar to Sander *et al.* [108], Julius *et al.* [60] use Lloyd iterations of growing and reseeding. But instead of looking for planar regions they search for a larger subset of developable surfaces, the so called "developable surfaces of constant slope," which are characterized by having a constant angle between the normal to the surface and some axis vector. They provide a metric to measure if a chart closely approximates such a surface, and use this metric in the chart growing and reseeding stages (Figure 3.9(c)).

4.1.1 Chart Packing

Chartification techniques raise an additional post-processing challenge. Following the parameterization of each individual chart, those charts need to be placed, or *packed*, in a common parameter domain. For efficient storage of the parameterized meshes, the packing has to be as compact as possible. The optimal packing problem is NP-hard [87], thus only heuristic or approximate packing algorithms exist. The Tetris algorithm [79] introduces charts one by one, searching for the best fit along the active-front of the charts packed so far (Figure 3.9(b)). Sander *et al.* [108] extend this algorithm, testing for more options (Figure 3.9(d)). Zhou *et al.* [138] use a similar approach but consider also non-square parameter domains (Figure 3.9(a)).

If discontinuities along chart boundaries are not a concern, one way to obtain a parameterization with very low distortion is to view each triangle as a single chart. In this case, the challenge of chartification shifts to finding compact packing [16].

When the shape of the charts is fixed, more efficient packing is possible. Tarini *et al.* [16] parameterize a 3D mesh over a set of square charts as described in Section 5.1. They then store texture as a collection of small square images, and are able support triangles spanning multiple charts and perform correct filtering across tile boundaries by storing a secondary index structure. Mesh vertices have 3D, rather than 2D, texture coordinates, mapping them to the space in the vicinity of a base domain formed by several cubes glued together. These 3D coordinates are interpolated for each pixel drawn for the mesh, and then a fragment program on the graphics card projects them onto a nearby cube face (identified using a small 3D texture) and reads the associated texture for the face.

4.2 Seam Cutting

It is possible to reduce the parameterization distortion without cutting the surface into separate patches by introducing multiple partial cuts or seams inside a single patch. This typically leads to shorter cuts than those created by segmentation. Piponi and Borshukov [95] generated such cuts manually using a network of edges.

McCartney *et al.* [86] and Sorkine *et al.* [120] perform parameterization and cutting simultaneously. They unfold the mesh vertices onto the plane one after the other, optimizing the local mapping. Whenever the distortion of the mapping reaches a threshold, Sorkine *et al.* [120] cut the mesh to reduce it. As a result they have a hard bound on the distortion, but can end up with long and complicated boundaries (Figure 4.1(a)). To measure distortion, Sorkine *et al.* [120] use the singular values γ and Γ of the per-triangle mapping (Section 2).

Gu *et al.* [41] use parameterization results to facilitate the cutting process. The authors first parameterize the surface using shape-preserving parameterization [32]. They then find the point of maximal parametric distortion on the mapping, and generate the shortest cut from the surface boundary to that point. They repeat the process until the distortion falls below a certain threshold (Figure 4.1(b)).

The *Seamster* algorithm [112, 117] considers the differential geometry properties of the surface, independent of a particular parameterization technique. It first finds regions of high Gaussian curvature on

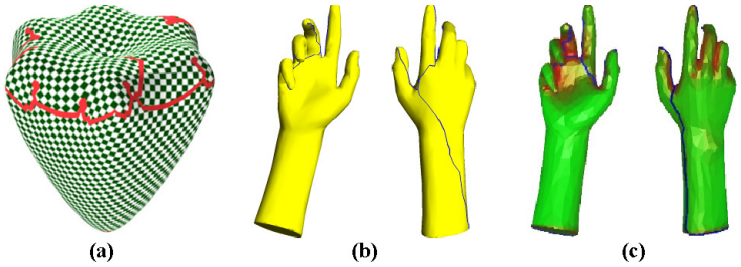


Fig. 4.1 Generating seams: (a) Sorkine *et al.* [120]; (b) geometry images [41]; and (c) seamster [117]. The color in (c) shows visibility – green is more visible, red is less. The Seamster cuts go across the palm which is considered more occluded than the back of the hand.

the surface and then uses a minimal spanning tree of the mesh edges to connect those. Finally, it cuts the mesh along the tree edges. While Sheffer [112] uses only the edge length criterion when generating the cuts, Sheffer and Hart [117] use a visibility metric as an edge weight when computing the minimal spanning tree. This way they are able to trace the cuts through the less visible parts of the surface hiding the potential cross-cut discontinuities in texture or other maps on the surface (Figure 4.1(c)). This algorithm was also used to cut the cow and camel in Figures 3.3 and 3.4.

While methods that segment meshes into multiple charts implicitly generate charts with disk topology, seam cutting methods, such as those of Sheffer and Hart [117] and Gu *et al.* [41] require an explicit pre-processing stage to convert surfaces with high genus into topological disks. The generation of minimal length cuts that convert a high genus surface into a topological disk is NP-hard [30]. Lazarus *et al.* [70] introduce a method for extraction of canonical schema, converting a high genus surface into a topological disk which in practice produces relatively short cuts. Erickson and Har-Peled [30] propose a handle cutting method which has some elegant theoretical guarantees but is complex to implement. A more practical approach is taken by Gu *et al.* [41], who trace a spanning graph of all the faces in the mesh and then prune this graph, obtaining a genus reducing cut. Ni *et al.* [90] smooth scalar functions over a mesh obtaining "fair" Morse functions that have few critical points and whose Morse complexes generate a small number of cuts for genus reduction.

5

Alternate Base Domains

Some applications are quite sensitive to discontinuities in the parameterization, or cannot tolerate them at all. In such cases, when the object to be parameterized is not a topological disk, it is worthwhile to use a different base domain for the parameterization. Examples of such domains that have been investigated include simplicial complexes (Section 5.1), spheres (Section 5.2), and periodic planar regions with transition curves (Section 5.3).

In addition, numerous applications of parameterization require cross-parameterization or inter-surface mapping between multiple models [66, 109]. Pair-wise mappings between models can be used for the transfer of different properties between the models, including straightforward ones, such as texture, and less obvious ones such as deformation and animation [122]. It can also be used for blending and morphing [3, 66, 109], as well as mesh completion and repair [67]. The most common approach for pair-wise mapping is to parameterize both models on a common base domain. Free-boundary planar parameterization is clearly unsuitable for this purpose. Instead, alternate domains such as a simplicial complex or a sphere are commonly used.

5.1 Simplicial and Quadrilateral Complexes

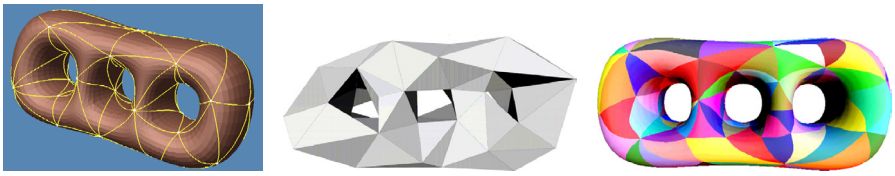
Historically, the most popular non-planar base domain has been a simplicial complex [48, 49, 63, 66, 71–73, 99, 100, 108]. A simplicial complex can be considered as just the connectivity part of a traditional triangle mesh: the sets of vertices, edges, and faces. Most applications typically use simplicial complexes representing two-manifolds with a boundary (an edge can only be adjacent to one or two faces) with a small number of elements. One method for obtaining such complexes is to simplify an original mesh. Once a suitable base mesh has been chosen, the original mesh is parameterized by assigning each of its vertices to a simplex of the base domain (vertex, edge, or face), along with barycentric coordinates inside it.

Early methods took a two-step approach to compute a parameterization; in the first step, elements of the fine mesh were assigned to faces of the base simplicial complex, while the second step would compute barycentric coordinates for these elements, usually using one of the fixed-boundary parameterization methods discussed in Section 3. These steps could be repeated, but typically not mixed. More recent methods, such as [63], try to perform both steps at the same time.

5.1.1 Computing Base Complexes

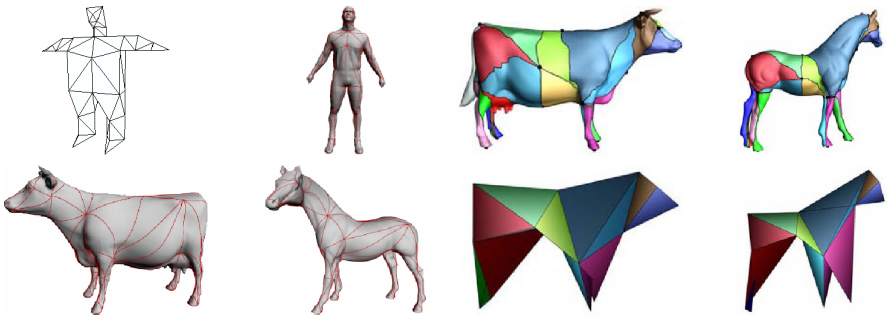
To obtain the simplicial complex, Eck *et al.* [28] grow Voronoi regions of faces from seed points and then use the dual triangulation (Figure 5.1(a)). The seed points are initially linked using shortest paths across mesh edges that provide the initial boundaries of the patches corresponding to base domain faces. To straighten each of these paths, the two adjacent patches are parameterized to a square. The path in question is then replaced with the diagonal of the square mapped onto the mesh surface.

Lee *et al.* [73] simplify the original mesh, keeping track of correspondences between the original vertices and the faces of the simplified mesh (Figure 5.1(b)). Others, like Guskov *et al.* [49] (Figure 5.1(c)) and Khodakovsky *et al.* [63] use clustering techniques to generate the patch connectivity and derive the base-mesh from it.

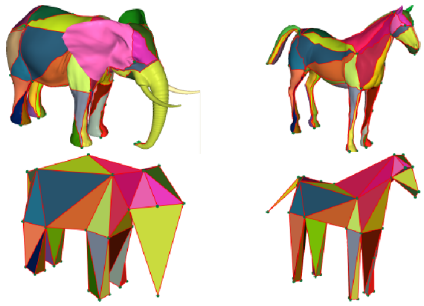


(a) Eck *et al.* [28] (b) Lee *et al.* [73] (c) Guskov *et al.* [49]

Fig. 5.1 Base mesh construction.



(a) Praun *et al.* [99] (b) Schreiner *et al.* [109]

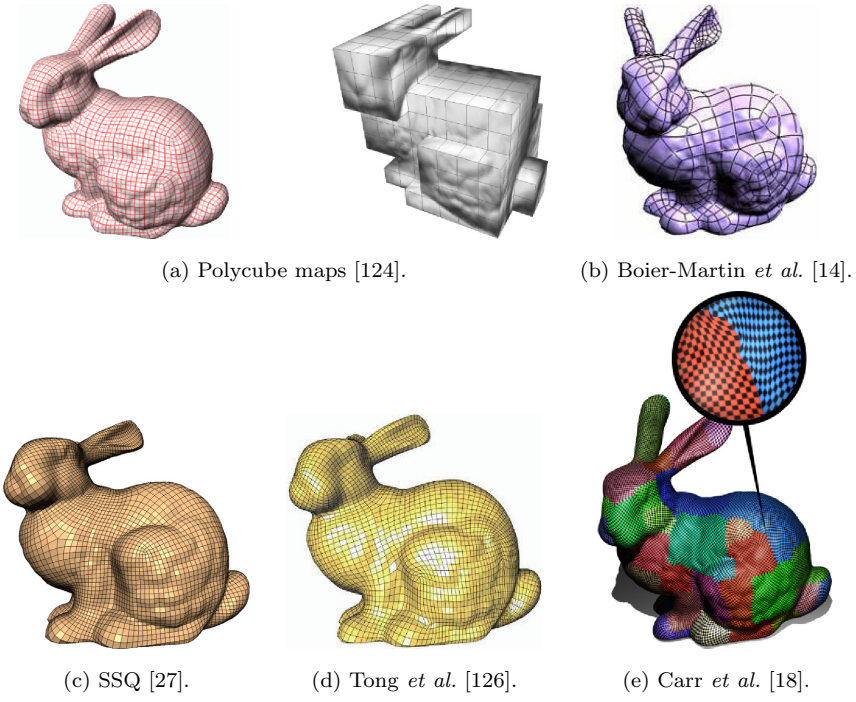


(c) Kraevoy and Sheffer [66]

Fig. 5.2 Consistent base-mesh creation.

The construction becomes more challenging when multiple models need to be parameterized on the same complex [66, 99, 109]. Praun *et al.* [99] partition a mesh into triangular patches, which correspond to the faces of a user given simplicial complex, by drawing a network of paths between user-supplied feature vertices that correspond to the vertices of the base-mesh (Figure 5.2(a)).

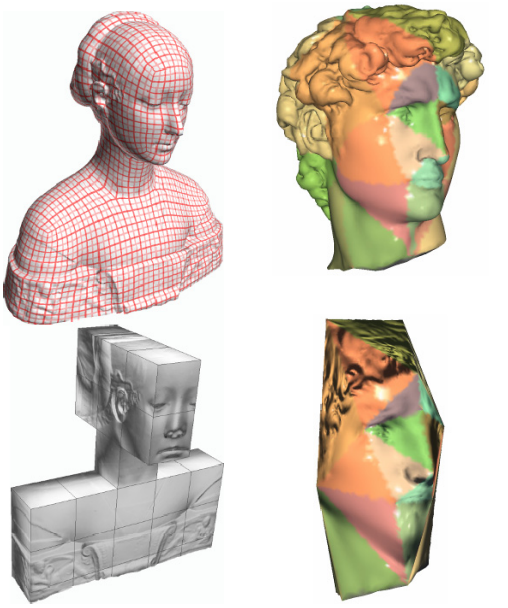
Kraevoy *et al.* [68] address a similar problem in the context of constrained planar parameterization (Section 6). Schreiner *et al.* [109], and Kraevoy and Sheffer [66] extend the methods of Praun *et al.* [99] and Kraevoy *et al.* [68] to construct the simplicial complex automatically, in parallel to the patch formation. The input to both methods includes a set of correspondences between feature vertices on the two input models. The methods use those as the vertices of the base complex. They simultaneously trace paths on the input meshes between corresponding pairs of vertices, splitting existing mesh edges if necessary (Figure 5.2(b) and 5.2(c)). Tarini *et al.* [124] were the first, to our knowledge, to use a quadrilateral base domain (Figures 5.3(a) and 5.4(a)). Such a domain is much more suitable for quadrilateral remeshing of the input surface and for spline fitting. Tarini *et al.* [124] generate the base domain manually.



(a) Polycube maps [124]. (b) Boier-Martin *et al.* [14].

(c) SSQ [27]. (d) Tong *et al.* [126]. (e) Carr *et al.* [18].

Fig. 5.3 Quadrilateral base domain construction and mapping.



Tarini *et al.* [124]. Kraevoy and Sheffer [66].

Fig. 5.4 Maps to base domains.

Boier-Martin *et al.* [14] first construct a coarse mesh using normal-based clustering of faces followed by spatially based clustering of the initially generated charts. This coarse mesh is then cleaned up and quadrangulated, yielding the base domain over which the input mesh is parameterized (Figure 5.3(b)).

Recently, Tong *et al.* [126] introduced a method for semi-automatic construction of curvature aligned quadrilateral base meshes. The method finds the umbilical points of the curvature field and then connects them along paths aligned with the principal curvature directions. They then proceed to find a global parameterization across the base domain as explained in Section 5.3 (Figure 5.3(d)).

Dong *et al.* [27] combine spectral analysis and Morse theory, computing the Morse complex of one of the eigenvectors of the Laplacian matrix (Equation (3.1)) weighted by the cotangent weights (Section 3.2). After smoothing, the Morse complex partitions the mesh into quad patches (Figure 5.3(c)). The constructed base domains and the

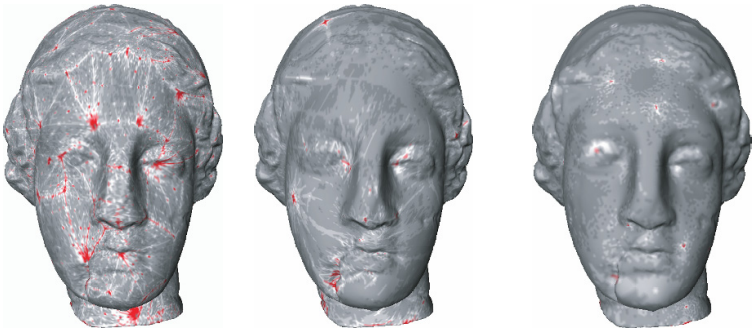
subsequent parameterization are typically not aligned with principal curvatures, an undesirable property when the parameterization is used for surface fitting or remeshing.

Similar to Sorkine *et al.* [120], Carr *et al.* [18] grow clusters outward from initial seed faces. To constrain cluster growth to a rectangular shape, a parameterization distance metric is employed which roughly approximates geodesic distance on the surface. The rectangular charts are then parameterized using shape-preserving parameterization [32], with the constraint that the parameterization is continuous on chart boundaries (Figure 5.3(e)).

5.1.2 Mapping to the Base Mesh

Once the discrete assignment to base domain faces has been done, the barycentric coordinates can be computed using fixed-boundary planar parameterization with the techniques from Section 3 [28, 49, 72]. Earlier methods computed the barycentric coordinates once, based on the initial assignment of the vertices to the base triangles (Figure 5.5(a) and 5.5(b)). More recent methods [63, 66, 67, 120] use a process where vertices can be reassigned between base faces.

Khodakovsky *et al.* [63] perform the vertex-to-patch assignment and coordinate relaxation in a single procedure, by letting vertices cross patch boundaries using transition functions (Figure 5.5(c)).



(a) MAPS [73]. (b) Guskov *et al.* [63]. (c) Khodakovsky *et al.* [63].

Fig. 5.5 Mappings to base, visualizing derivative magnitudes of the tangent field (gray: small; red: large). The figures come from Khodakovsky *et al.* [63].

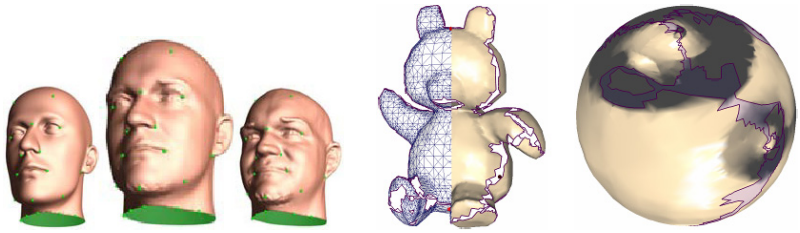
A transition function expresses the barycentric coordinates of a vertex with respect to a base domain face as barycentric coordinates for a neighboring base domain face. For this procedure, only the images of the base domain vertices needs to be fixed, rather than the edges as well as in the previous methods. The authors relax the base domain vertices separately, prompting a new run of the main relaxation. In practice, this cycle is repeated only very few times. The implementation sometimes needs to discard some relaxation results when mesh vertices moved around base domain vertices end up with barycentric coordinates that are invalid for all the base domain faces around that vertex.

Tarini *et al.* [124] and Kraevoy and Sheffer [66, 67] fix the boundary of a group of base mesh faces, update the barycentric coordinates in the interior, and then possibly re-assign some vertices to different faces inside the group. The methods differ in the grouping they use and the choice of parameterization technique used for the barycentric coordinates computation (Figure 5.4).

Schreiner *et al.* [109] never compute an explicit map between the full-resolution objects and the base domain. Instead, they alternate the role of base domain between the two meshes, at various complexity levels in a multi-resolution representation. They progressively refine each mesh by adding new vertices and relaxing their location using a stretch-based metric measured on a temporary planar unfolding of their neighborhoods.

Most of the methods listed above assume that the input meshes are closed. Schreiner *et al.* [109] handle meshes with boundaries by mapping the boundaries to edges of the base complex and the corresponding holes to a subset of its faces (Figure 5.6(a)). Kraevoy and Sheffer [67] support parameterization of meshes with both holes and multiple components (Figure 5.6(b)). They allow free-boundary parameterization of the holes and the outer boundaries of the components by introducing a *virtual* triangulation of the holes and gaps, and updating this triangulation during the iterative parameterization process. This parameterization method is particularly suitable for mesh completion operations.

The modern parameterization methods that use simplicial or quadrilateral complexes, as described above, typically produce



(a) Schreiner *et al.* [109]. (b) Kraevoy and Sheffer [67].

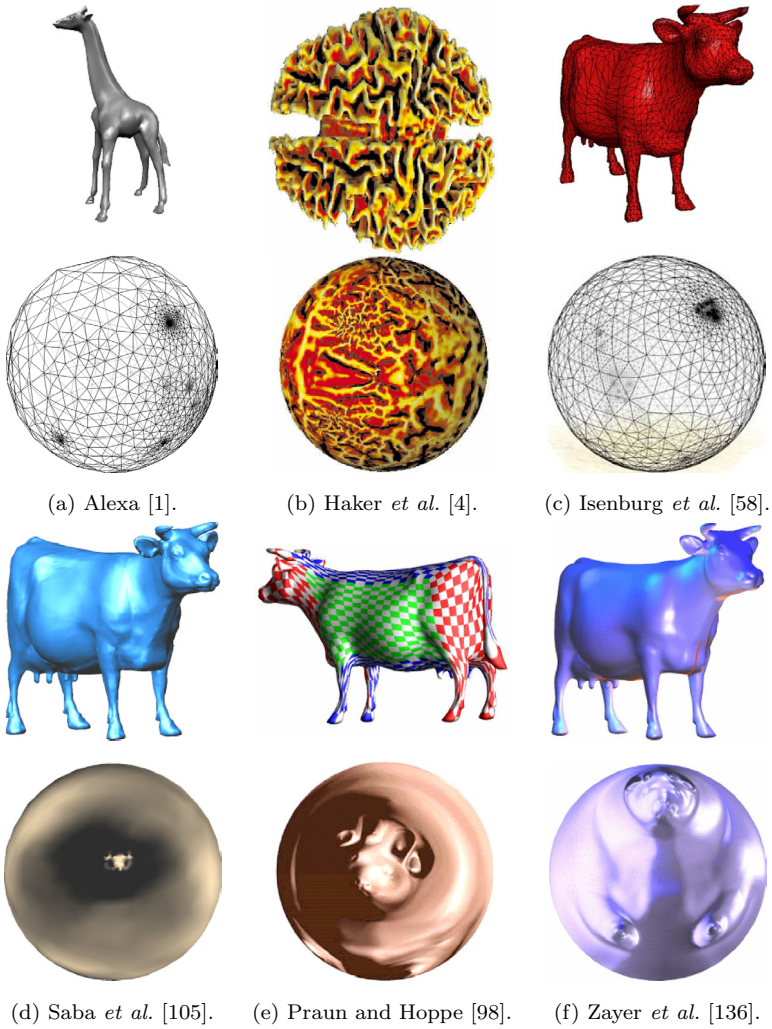
Fig. 5.6 (left) Schreiner *et al.* [109] parameterize connected models with boundaries. The center model is 50% morph of the side ones. (right) Kraevoy and Sheffer [67] parameterize meshes with multiple components, such as the teddy mapped here to a sphere.

parameterizations which are visually smooth everywhere except in the vicinity of irregular vertices of the base meshes.

5.2 The Unit Sphere

The big advantage of the spherical domain over the planar one is that it allows for seamless, continuous parameterization of genus-0 models, and there are a large number of such models in use. Thus, the spherical domain has received much attention in the last few years, with several papers published about this topic [38, 50, 98, 105, 106]. Some rigorous theory is being developed, getting close to the level of understanding we have of planar parameterizations.

One attractive approach for spherical parameterization is to extend the barycentric, convex boundary planar methods to the sphere. Several methods [1, 42, 45, 64] used Gauss–Seidel iterations to obtain such parameterization. They start by computing an initial guess and then moving the vertices one at a time, first computing a 3D position for the vertex using a barycentric formulation [28, 32], and then projecting the vertex to the unit sphere (Figure 5.7(a)). Isenburg *et al.* [58] split the mesh in two, map the cut onto a great circle and embed each half-mesh onto a hemisphere using a modified Tutte procedure (Figure 5.7(c)). Regrettably, as proven by Saba *et al.* [105] "projected" Gauss–Seidel iterations decrease the residual for only a finite number of iterations. As the result approaches a bijective solution, the scheme ultimately becomes unstable, the residual increases, and the system collapses to



(a) Alexa [1]. (b) Haker *et al.* [4]. (c) Isenburg *et al.* [58].

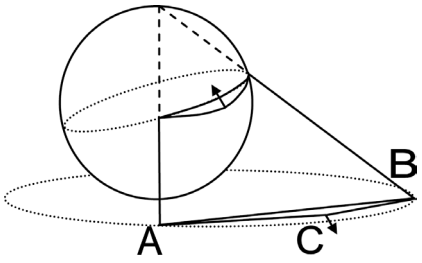
(d) Saba *et al.* [105]. (e) Praun and Hoppe [98]. (f) Zayer *et al.* [136].

Fig. 5.7 Spherical parameterizations.

a degenerate solution. Saba *et al.* [105] note that this behavior is independent of step size.

Birkholz [9] uses a hierarchical method which works with the spherical angles of the embedding. The hierarchical approach stabilizes the convergence, but is able to obtain only an approximate solution.

Haker *et al.* [50] compute a planar parameterization of the mesh first and then use the stereographic projection to obtain the spherical mapping (Figure 5.7(b)). This approach works quite well in practice even for dense meshes, however it does not offer any theoretical guarantees since the stereographic projection is bijective only for the continuous case, and can produce triangle flips in the discrete case. A simple proof by example of this statement can be obtained by imagining the great circle supporting the edge AB of a mapped spherical triangle ABC, below. The (continuous) stereographic projection maps this great circle to a circle in the original plane. The third vertex C can be perturbed in the plane to cross from the interior to the exterior of the circle, without changing the triangle orientation. The spherical triangle ABC will flip however as a result of this perturbation, as the image of C on the sphere will cross from one side to the other of the spherical edge AB.



Gotsman *et al.* [38] showed how to correctly generalize the method of barycentric coordinates, with all its advantages, to the sphere. The generalization is based on results from spectral graph theory due to Colin de Vérdiere [22] and extensions due to Lovasz and Schrijver [83]. They provide a quadratic system of equations which is a spherical equivalent of the barycentric formulation. The authors do not provide an efficient way to solve the resulting system, and thus their method is limited to very small meshes. Saba *et al.* [105] introduce a method for efficiently solving the system, by providing a good initial guess and using a robust solver (Figure 5.7(d)). First, similar to Isenburg *et al.* [58], they partition the mesh in two, and embed each half on a hemisphere using a planar parameterization followed by a stereographic projection. They

then use a numerical solution mechanism which combines Gauss–Seidel iteration with nonlinear minimization to obtain the final solution.

Sheffer *et al.* [116] extend the ABF method [113] to spherical embeddings, formulating the problem in terms of spherical angles. Regrettably, it appears that the spherical formulation is numerically much less stable than its planar equivalent. Thus the method seems impractical for large meshes.

An efficient and bijective alternative is suggested by multi-resolution techniques. These methods obtain an initial guess by simplifying the model until it becomes a tetrahedron (or at least, convex), trivially embed it on the sphere, and then progressively add back the vertices [98, 111]. Shapiro and Tal [111] compute the embedding using purely topological operations and do not attempt to minimize any type of distortion. Praun and Hoppe [98] obtain a spherical parameterization by alternating refining steps that add vertices from a multi-resolution decomposition of the object with relaxation of single vertex locations inside their neighborhoods. The relaxation is aimed to minimize the stretch metric of the parameterization and is guaranteed to maintain a valid embedding (Figure 5.7(e)).

Zayer *et al.* [136] introduce a spherical parameterization method which takes user prescribed poles into account (Figure 5.7(f)). They then cut the mesh along a date line connecting these poles. With the mesh then topologically equivalent to a disk, an initial parameterization is defined over a rectangular domain by solving a Laplace equation in curvilinear coordinates. The parameterization distortion is reduced using a variant of quasi-harmonic maps [134]. Finally, they reduce the distortion along the date line by performing tangential Laplacian smoothing.

5.3 Discrete Differential One-Forms

The classical approach for parameterizing models of arbitrary topology was to first construct a base-complex and then parameterize the mesh onto the complex (Section 5.1). Recently, several methods [46, 62, 102, 126], have proposed implicitly creating a parameterization by solving for differential one-forms. Instead of associating (u,v)

coordinates with vertices as in typical planar parameterization, they associate planar vectors (du, dv) with the edges of the mesh, computing a gradient field, or *one-form*. These parameterizations can be converted to a special form of planar parameterization by fixing a vertex, then walking around to other vertices and adding edge vectors. The method guarantees that inside a topological disk region, the same coordinates are obtained for each vertex regardless of the path used to walk to it from the source vertex. Genus-1 surface can be mapped to an infinite plane using a multi-periodic function by tiling the plane using a *modular parallelogram* – a fundamental domain bounded by four curves, parallel two by two. For higher genus, applying this method directly will cover the plane multiple times. Such parameterizations are guaranteed to be continuous everywhere, except at a small number of singular points, and are therefore sometimes referred to as globally continuous parameterizations.

The first method for global periodic parameterization was proposed by Gu and Yau [46] (Figure 5.8(a)). After computing a holomorphic one-form, corresponding to a global conformal parameterization, the authors propose constructing a global conformal atlas for the surface, by separating the object handles and mapping each of them to a modular parallelogram. Each parallelogram can have

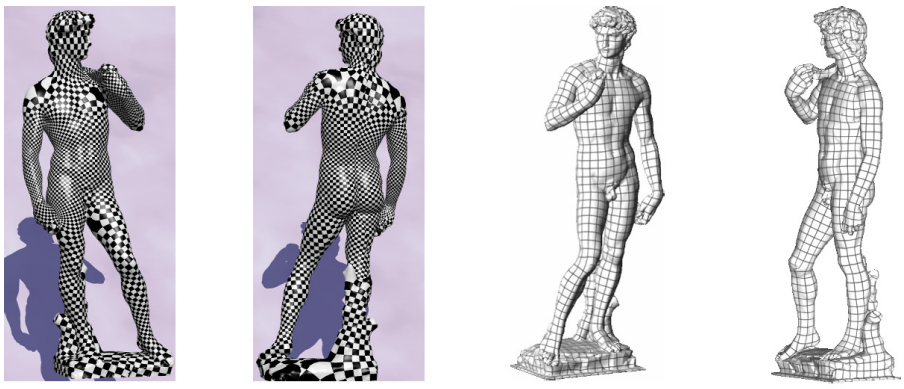


Fig. 5.8 (a) Global conformal surface parameterization [46], (b) periodic global parameterization [102].

interior transition curves that conformally connect the parameterization to corresponding curves inside other modular parallelograms. The images on the mesh of these transition curves separate the handles.

The theory behind this parameterization comes from an elegant connection to Riemann geometry [89]. In practice, computing the parameterization involves solving linear systems with three types of constraints: harmonicity, closedness, and duality. The harmonicity equations are similar to the ones used by methods in Section 3: the sum of vectors for edges incident to a vertex, weighed by the cotangent harmonic weights, is zero. The closedness equations state that the sum of the three vectors for the edges of each face is zero (a "gradient field" has no divergence). The duality conditions replace the boundary conditions in traditional parameterizations; they impose fixed values for the integral of the field on the closed loops forming the homology basis of the surface (the $2g$ curves that *cut open* the handles of the mesh, where g is the genus of the surface). Several systems are solved, in each system the integral across one of the curves is constrained to 1, and all the others to 0.

The same authors present a faster, multi-resolution method [44], applications to mesh compression [43], and brain surface mapping [42]. Gu *et al.* [44] find the unique locations of the singular points that satisfy conformality. They also solve for the optimal conformal transformation that minimizes global stretch. Since the number of singular points remains constant, the resulting parameterizations often still exhibit significant stretch. The methods do not guarantee bijectivity but perform well in practice.

Gortler *et al.* [37] provide further theoretical insight into the applications of one-forms to parameterizing meshes. They develop a discrete counterpart of the Hopf–Poincaré index theorem. This theorem generalizes the notion of singular points mentioned above and allows their splitting and merging. It states that for a surface of genus g, the indices of all singular points sum to $2g-2$, where the index of a singular point corresponds to its multiplicity. Their results include a simpler proof of Tutte's theorem and a proof that the method applied to genus-1 objects indeed produces a locally bijective embedding (no triangle

folds), for any set of asymmetric positive weights. The proof extends to non-triangular meshes as well.

Motivated by geometry processing applications, Ray *et al.* [102] incorporate more geometric information into the problem setting (Figure 5.8(b)). Given the curvature tensor of the surface, their method computes two piecewise linear periodic functions, aligned with the minimal and maximal curvatures. Together these two functions define a smooth parameterization almost everywhere, except in the vicinity of the singular points of the tensor field. The authors demonstrate how to extract a quadrilateral chart layout from this periodic parameterization. The method can construct quasi-isometric parameterizations at the expense of introducing additional singular points in non-developable regions where the curl of the input field is non-zero. They also propose a curvature-adapted parameterization method that minimizes the curl and removes those additional singular points by adaptively scaling the parameterization. The authors demonstrate that the method is well suited for quadrilateral remeshing and surface fitting.

Kharevych *et al.* [62] extend their method of parameterization using circle patterns to global parameterization. They observe that in angle space formulation the only difference between parameterizing the mesh boundary and its interior mesh is in the constraint imposed on the sum of vertices in the interior, and that it is possible to define a global parameterization by specifying an unconnected subset of mesh vertices as boundary vertices. Thus they first compute a solution in angle space with a set of *cone singularity* vertices specified by the user as boundary. For planar parameterization, to perform the angle-to-uv conversion they later compute edge paths between these vertices. The obtained parameterization is globally continuous up to translation and rotation, everywhere except at the cone singularities. The proposed approach can be directly applied to other angle-space methods such as ABF/ABF++.

Tong *et al.* [126] compute a global parameterization by expressing the differential one-form system back in terms of the associated potential fields (the zero-form, or uv variables per vertex) augmented with a set of line singularities. This singularity graph is similar to a quadrilateral base domain (Figure 5.4) used by the methods in Section 5.1,

and can be either user-provided or generated semi-automatically from the mesh curvature.

5.4 Hyperbolic Plane

Objects of genus n can be mapped to an infinite tiling of the hyperbolic disk with $4n$-sided regular polygons. Straight lines from Euclidean geometry correspond to arcs that intersect the main hyperbolic disc at 90°. The fundamental domain is a regular hyperbolic polygon centered at the origin, where each opposing pair of edges corresponds to one of the $2n$ loops that cut open the handles of the object. All the vertices of the polygon correspond to a single point on the object, where all the loops meet, which is also the only intersection point between any two loops. The sum of angles of the polygon is 2π (which is possible for a hyperbolic polygon with more than three sides). To produce the tiling, the polygon is repeatedly "flipped" over an edge, using linear fractional transforms: $(z) \rightarrow (\mathrm{az} + \mathrm{b}/\mathrm{cz} + \mathrm{d})$, where z is a point inside the disk, represented as a complex number.

A parameterization based on hyperbolic geometry was proposed by Ferguson and Rockwood [31], and Rockwood and Park [103]. Grimm and Hughes [40] use a hyperbolic parameterization as the starting point for creating a manifold: an atlas with several overlapping charts and transition functions between them. Each chart corresponds to one of the

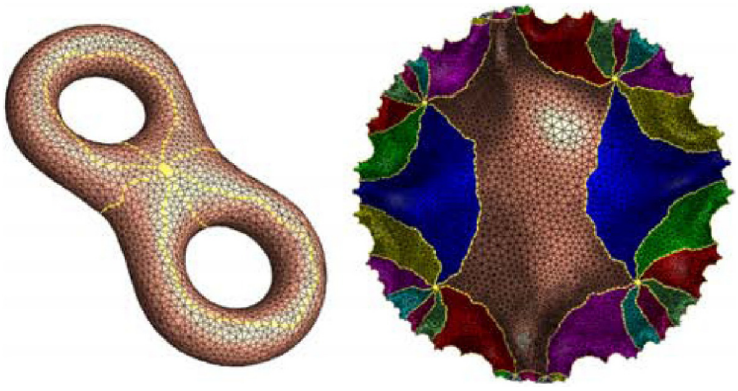


Fig. 5.9 Hyperbolic parameterization [59].

elements of the base domain. In the case of the original un-subdivided regular polygon: one face, one (shared) vertex, and $2n$ (pairs of) edges. Each chart overlaps charts of adjacent elements, resulting in a transition function, and does not overlap any charts of non-adjacent elements. Jin *et al.* [59] provide a practical algorithm for computing hyperbolic parameterization for meshes of arbitrary genus (Figure 5.9).

6

Constraints

Sometimes a parameterization needs to accommodate user input, in the form of correspondences between vertices of the mesh, or curves traced on the surface, and locations in the domain. The most important application of constrained parameterization is texture mapping 3D models from photographs [29, 68, 76]. Zhou *et al.* [139] go further and allow the user to combine several images to produce a complete texture for a mesh. Constrained parameterization can also be used to hide cross-seam discontinuities [68, 139].

Constraints are also used to establish maps with feature correspondence between different 3D objects [66, 109]. Such maps between meshes allow morphing, transferring of textures and other attributes between meshes, and principal component analysis of similar models in a database. Constrained parameterization methods can be grouped into two large categories, based on whether they treat this user input as "suggestions" or "help," or rather, hard constraints to be satisfied exactly.

Soft Constraints Methods based on energy minimization can accommodate soft constraints by adding a quadratic term to the energy

Fig. 6.1 Soft constraints [76]. Mapping the tiger texture (a) onto the face in (b) yields (c) and (d).

function, measuring the distance between the constraint features in the current configuration and their desired location [76] (see Figure 6.1). Such constraints work reasonably well in practice, and can be solved efficiently, since they only add linear terms to the energy, but sometimes break theoretical guarantees about the original parameterization method, such as bijectivity. Allen *et al.* [3] deform a template mesh to match a series of laser scans of human bodies by softly matching a set of about 70 marker constraints and a dense set of automatically computed point correspondences between the surfaces. Sumner and Popović [122] use similar ideas to transfer deformation (and whole animation sequences) from one mesh to another, though they do not extract an explicit one-to-one mapping between surfaces. Note that in these last two papers the energy being minimized takes into account more than just the difference between the actual and desired locations of a constraint, and actually softly matches a whole affine transformation matrix defining the surfaces around each constraint. These methods typically work well when the constraints do not require significant changes in the output compared to the unconstrained result. In addition, the methods no longer guarantee that the resulting parameterization will remain bijective.

Hard Constraints Applications such as hiding texture discontinuities along seams in the parameterization [68, 139] require hard constraints to achieve perfect alignment of the texture along the seams. In

addition, hard constraints can be enforced while preserving parameterization bijectivity.

It is possible to enforce constraints by adding them into a regular parameterization formulation using Lagrange multipliers [25]. This approach allows constraints to be defined on points inside triangles and on arbitrary line segments (the vertices of the triangulation can be constrained more easily by taking the corresponding variables out of the system). However, it is easy to show that for a given mesh connectivity not every set of constraints can be satisfied. Thus methods like this that preserve the mesh connectivity will fail to generate bijective parameterizations for many inputs.

It is known that an embedding can always be found if the mesh is enriched by adding a number of additional *Steiner* vertices [91], Regrettably, finding the minimal number of vertices that have to be added is NP-Hard [91].

Methods that enforce hard constraints therefore modify the provided meshes by inserting a few new Steiner vertices [29, 68] or by complete remeshing [99].

Eckstein *et al.* [29] introduce a method that enforces "hard" (exact) positional constraints by deforming an existing embedding while adding a number of Steiner vertices (Figure 6.2). Theoretically, this method can handle large sets of constraints but is extremely complicated.

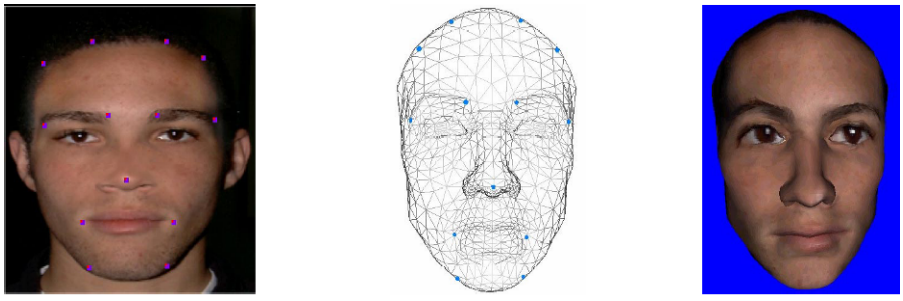


Fig. 6.2 Eckstein *et al.* [29] – Mapping a (a) texture of a face to a (b) mesh of a face. Result is in (c).

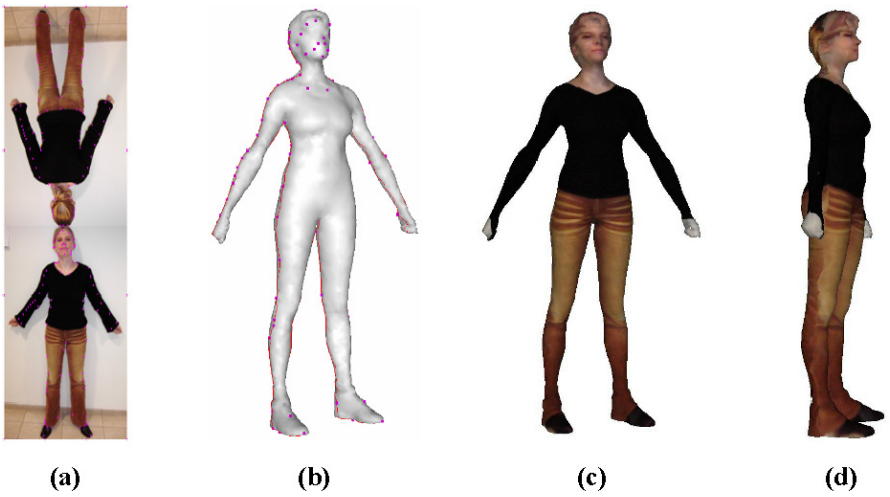


Fig. 6.3 Mapping a photograph onto a model using Matchmaker [68]. Pink dots in (a) and (b) represent constraints. (c) and (d) show various views of the texture mapped model.

Instead, Praun *et al.* [99] and Kraevoy *et al.* [68] compute the parameterization by establishing coarse patch correspondences between the input and the parameter domain (Figure 6.3). The provided feature points on the input model and the parameter domain are connected using a network of curves that partition the surface into patches that are then parameterized while trying to maintain continuity and smoothness between them. The curve tracing process is guided by a set of topological rules that ensure that the resulting patches will be consistent between the object(s) being parameterized and the domain, and by heuristics that try to guide them on the correct side of constraints and prominent model features (such as limbs), avoiding awkward parameterization "swirls." Praun *et al.* [99] use a user provided triangulation of the features on the parameter domain, and enforce it on the model, while Kraevoy *et al.* [68] compute the two triangulations simultaneously, solving a more challenging problem. Continuity and smoothness between patches can be obtained by parameterizing or relaxing the parameterization of several patches simultaneously.

Zhou *et al.* [139] allow the user to combine several images to produce texture for a mesh, by assigning some surface patches to different

(a) (b) (c)

Fig. 6.4 Texturing Buddha from textures of multiple objects [139]. From left to right: input images, generated texture atlas and mapping results.

images, as well as using in-painting techniques to create texture for any unassigned transition patches between them (Figure 6.4). In addition to geometric smoothness of the map, Zhou *et al.* [139] take into account the continuity of the texture signal being applied since it may come from different sources for two neighboring patches.

7

Conclusions

In this paper, we reviewed recent mesh parameterization techniques, as well as related mesh processing tools such as mesh segmentation and seam cutting. We focused on the more practical aspects of using parameterization for computer graphics applications, discussing method properties such as speed, robustness, and ease of implementation.

We would like to note that judging by the amount of active research on parameterization, no survey on the topic can, realistically, be truly complete, and ours is no exception. We are sure to have missed at least a few of the related techniques introduced in the last couple of years.

Finally, we observe that in recent years many links have emerged between parameterization and other mesh processing techniques, such as mesh editing and surface approximation which are not addressed by this survey. We also did not address parameterization of other types of surface representations, such as free-form surfaces and point-based surfaces, which provide challenges of their own, and for which there is a separate, vast body of work.

We believe that while there are numerous methods for surface parameterization available, many challenges remain. There is room for further research on topics such as global-parameterization techniques,

techniques that address complex constraints, and segmentation. There are also many open questions in terms of finding suitable numerical tools and better understanding the theoretical aspects of parameterization techniques as well as convergence properties for the numerical tools used.

Acknowledgments

The authors would like to thank the reviewers for their comments and Ciarán Llachlan Leavitt for her help in editing the paper. Sheffer and Rose gratefully acknowledge the research support of NSERC and IBM.

References

[1] M. Alexa, "Merging polyhedral shapes with scattered features," *The Visual Computer*, vol. 16, no. 1, pp. 26–37, 2000.

[2] M. Alexa, "Recent advances in mesh morphing," *Computer Graphics Forum*, vol. 21, no. 2, pp. 173–196, 2002.

[3] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: Reconstruction and parameterization from range scans," in *ACM SIGGRAPH 2003*, pp. 587–594, 2003.

[4] P. Alliez and C. Gotsman, "Recent advances in compression of 3D meshes," in *Advances in Multiresolution for Geometric Modelling*, pp. 3–26, 2005.

[5] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, *Recent advances in remeshing of surfaces*. Springer, 2005.

[6] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape completion and animation of people," in *ACM SIGGRAPH 2005*, 2005.

[7] C. Bennis, J.-M. Vézien, and G. Iglésias, "Piecewise surface flattening for nondistorted texture mapping," in *ACM SIGGRAPH '91*, pp. 237–246, 1991.

[8] H. Biermann, I. Martin, F. Bernardini, and D. Zorin, "Cut-and-paste editing of multiresolution surfaces," in *ACM SIGGRAPH 2002*, pp. 312–321, 2002.

[9] H. Birkholz, "Shape-preserving parametrization of genus 0 surfaces," in *Proceedings of Winter Conference on Computer Graphics (WSCG)*, 2004.

[10] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating faces in images and video," *Computer Graphics Forum (EUROGRAPHICS 2003)*, vol. 22, no. 3, pp. 641–650, 2003.

[11] V. Blanz, K. Scherbaum, T. Vetter, and H.-P. Seidel, "Exchanging faces in images," in *EUROGRAPHICS 2004*, 2004.

[12] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *ACM SIGGRAPH 1999*, 1999.

[13] J. F. Blinn, "Simulation of wrinkled surfaces," in *ACM SIGGRAPH '78*, pp. 286–292, 1978.

[14] I. Boier-Martin, H. Rushmeier, and J. Jin, "Parameterization of triangle meshes over quadrilateral domains," in *Eurographics Symposium on Geometry Processing 2004*, pp. 197–208, 2004.

[15] P. Bowers and M. Hurdal, "Planar conformal mappings of piecewise flat surfaces," in *Visualization and Mathematics III*, pp. 3–34, Springer Verlag, 2003.

[16] N. Carr and J. Hart, "Meshed atlases for real-time procedural solid texturing," *ACM Transactions on Graphics*, vol. 21, no. 2, April 2002.

[17] N. Carr and J. Hart, "Painting detail," in *ACM SIGGRAPH 2004*, 2004.

[18] N. Carr, J. Hoberock, K. Crane, and J. Hart, "Rectangular multi-chart geometry images," in *Proceedings of Symposium on Geometry Processing (SGP)*, 2006.

[19] F. R. K. Chung in *Spectral Graph Theory*, Providence, RI: American Mathematical Society, 1997.

[20] U. Clarenz, N. Litke, and M. Rumpf, "Axioms and variational problems in surface parameterization," *Computer Aided Geometric Design*, vol. 21, no. 8, 2004.

[21] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," in *ACM SIGGRAPH 2004*, pp. 905–914, 2004.

[22] Y. Colin de Verdière, "On a new graph invariant and a criterion for planarity," in *Graph structure Theory/Contemporary Mathematics*, pp. 137–147, 1990.

[23] C. Collins and K. Stephenson, "A circle packing algorithm," in *Computational Geometry: Theory and Applications 25*, pp. 233–256, 2003.

[24] P. Degener, J. Meseth, and R. Klein, "An adaptable surface parameterization method," in *Proceedings, 12th International Meshing Roundtable*, pp. 201–213, 2003.

[25] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic parameterizations of surface meshes," *Computer Graphics Forum*, vol. 21, pp. 209–218, 2002.

[26] M. do Carmo, "Differential geometry of curves and surfaces," Prentice Hall, 1976.

[27] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart, "Spectral surface quadrangulation," in *ACM SIGGRAPH 2006*, 2006.

[28] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," in *ACM SIGGRAPH 95*, pp. 173–182, 1995.

[29] I. Eckstein, V. Surazhsky, and C. Gotsman, "Texture mapping with hard constraints," in *Eurographics 2001*, pp. 95–104, 2001.

[30] J. Erickson and S. Har-Peled, "Optimally cutting a surface into a disk," *Discrete Computational Geometry*, vol. 31, no. 1, pp. 37–59, 2004.

[31] H. Ferguson and A. Rockwood, "Multiperiodic functions for surface design," *Computer Aided Geometric Design*, vol. 10, no. 3, pp. 315–328, 1993.

[32] M. Floater, "Parameterization and smooth approximation of surface triangulations," *Computer Aided Geometric Design 1997*, vol. 14, no. 3, pp. 231–250, 1997.
[33] M. Floater, "Mean value coordinates," *Computer Aided Geometric Design*, vol. 20, no. 1, pp. 19–27, 2003.
[34] M. Floater and K. Hormann, "Parameterization of triangulations and unorganized points," *Tutorials on Multiresolution in Geometric Modelling*, 2002.
[35] M. Floater and K. Hormann, "Surface parameterization: A tutorial and survey," in *Advances in Multiresolution for Geometric Modelling*, pp. 157–186, 2005.
[36] M. Garland, A. Willmott, and P. Heckbert, "Hierarchical face clustering on polygonal surfaces," in *Proceedings of ACM Symposium on Interactive 3D Graphics 2001*, 2001.
[37] S. J. Gortler, C. Gotsman, and D. Thurston, "Discrete one-forms on meshes and applications to 3D mesh parameterization," *Computer Aided Geometric Design*, 2005.
[38] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of spherical parameterization for 3D meshes," in *ACM SIGGRAPH 2003*, pp. 358–364, 2003.
[39] Graphite http://www.loria.fr/∼levy/Graphite/index.html, 2003.
[40] C. Grimm and J. Hughes, "Parameterizing N-holed Tori," in *The Mathematics of Surfaces IX*, 2003.
[41] X. Gu, S. Gortler, and H. Hoppe, "Geometry images," in *ACM SIGGRAPH 2002*, pp. 356–361, 2002.
[42] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau, "Genus zero surface conformal mapping and its application to brain surface mapping," *IEEE Transaction on Medical Imaging*, vol. 23, no. 7, 2004.
[43] X. Gu, Y. Wang, and S.-T. Yau, "Geometric compression using Riemann surface structure," in *Communication in Information and Systems*, 2003.
[44] X. Gu, Y. Wang, and S.-T. Yau, "Multiresolution computation of conformal structures of surfaces," *Journal of Systemics, Cybernetics and Informatics*, vol. 1, no. 6, 2004.
[45] X. Gu and S.-T. Yau, "Computing conformal structures of surfaces," *Communications in Information and Systems*, vol. 2, no. 2, pp. 121–146, 2002.
[46] X. Gu and S.-T. Yau, "Global conformal surface parameterization," *Symposium on Geometry Processing*, pp. 127–137, 2003.
[47] I. Guskov, "An anisotropic mesh parameterization scheme," in *Proceedings of the 11th International Meshing Roundtable*, pp. 325–332, 2002.
[48] I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens, "Hybrid meshes: Multiresolution using regular and irregular refinement," in *ACM Symposium on Computational Geometry 2002*, pp. 264–272, 2002.
[49] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder, "Normal meshes," in *ACM SIGGRAPH 2000*, pp. 95–102, 2000.
[50] S. Haker, S. Angenent, S. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, "Conformal surface parametrization for texture mapping," *IEEE TVCG*, vol. 6, no. 2, pp. 181–189, 2000.

[51] Y. He, X. Gu, and H. Qin, "Rational spherical splines for genus zero shape modeling," in *Proceedings of IEEE International Conference on Shape Modeling and Applications 2005*, 2005.

[52] H. Hoppe and E. Praun, "Shape compression using spherical geometry images," in *Advances in Multiresolution for Geometric Modeling*, pp. 3–26, 2005.

[53] K. Hormann, "Fitting free form surfaces," in *Principles of 3D Image Analysis and Synthesis*, 2000.

[54] K. Hormann and G. Greiner, "MIPS: An efficient global parametrization method," in *Curve and Surface Design*, pp. 153–162, 2000.

[55] K. Hormann, G. Greiner, and S. Campagna, "Hierarchical parametrization of triangulated surfaces," *Vision, Modeling, and Visualization*, pp. 219–226, 1999.

[56] M. K. Hurdal, P. L. Bowers, K. Stephenson, D. W. L. Sumners, K. Rehm, K. Schaper, and D. A. Rottenberg, "Quasi-conformally flat mapping the human cerebellum," in *Medical Image Computing and Computer-Assisted Intervention 1679*, pp. 279–286, 1999.

[57] T. Igarashi and D. Cosgrove, "Adaptive unwrapping for interactive texture painting," in *ACM Symposium on Interactive 3D Graphics*, pp. 209–216, 2001.

[58] M. Isenburg, S. Gumhold, and C. Gotsman, "Connectivity shapes," in *Proceedings of IEEE Visualization*, pp. 135–142, 2001.

[59] M. Jin, F. Luo, and X. Gu,, "Computing surface hyperbolic structure and real projective structure, solid and physics modeling," in *Proceedings Solid and Physics Modeling*, 2006.

[60] D. Julius, V. Kraevoy, and A. Sheffer, "D-charts: Quasi-developable mesh segmentation," *Computer Graphics Forum (Proceedings of Eurographics 2005)*, vol. 24, no. 3, pp. 981–990, 2005.

[61] Z. Karni, C. Gotsman, and S. Gortler, "Free-boundary linear parameterization of 3D meshes in the presence of constraints," in *Proceedings of Shape Modeling and Applications 2005*, 2005.

[62] L. Kharevych, B. Springborn, and P. Schröder, "Discrete conformal mappings via circle patterns," *ACM Transactions on Graphics*, vol. 25, no. 2, 2006.

[63] A. Khodakovsky, N. Litke, and P. Schröder, "Globally smooth parameterizations with low distortion," in *ACM SIGGRAPH 2003*, pp. 350–357, 2003.

[64] L. P. Kobbelt, J. Vorsatz, U. Labisk, and H.-P. Seidel, "A shrink-wrapping approach to remeshing polygonal surfaces," in *Proceedings of Eurographics*, 1999.

[65] G. Kos and T. Varady, "Parameterizing complex triangular meshes," in *Curve and Surface Design*, pp. 265–274, 2003.

[66] V. Kraevoy and A. Sheffer, "Cross-parameterization and compatible remeshing of 3D models," in *ACM SIGGRAPH 2004*, 2004.

[67] V. Kraevoy and A. Sheffer, "Template based mesh completion," in *Proceedings of Symposium on Geometry Processing (SGP)*, 2005.

[68] V. Kraevoy, A. Sheffer, and C. Gotsman, "Matchmaker: Constructing constrained texture maps," in *ACM SIGGRAPH 2003*, pp. 326–333, 2003.

[69] U. Labsik, K. Hormann, and G. Greiner, "Using most isometric parametrizations for remeshing polygonal surfaces," in *Geometric Modeling and Processing*, 2000.

[70] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust, "Computing a canonical polygonal schema of an orientable triangulated surface," in *Proceedings of the 17th Annual Symposium on Computational Geometry*, (Medford, Massachusetts, United States), 2001.

[71] A. Lee, D. Dobkin, W. Sweldens, and P. Schröder, "Multiresolution mesh morphing," in *ACM SIGGRAPH '99*, pp. 343–350, 1999.

[72] A. Lee, H. Hoppe, and H. Moreton, "Displaced subdivision surfaces," in *ACM SIGGRAPH 2000*, 2000.

[73] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," in *ACM SIGGRAPH '98*, pp. 85–94, 1998.

[74] Y. Lee, H. S. Kim, and S. Lee, "Mesh parameterization with a virtual boundary," *Computers and Graphics*, vol. 26, no. 5, pp. 677–686, 2002.

[75] J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe, "Real-time fur on arbitrary surfaces," in *Symposium on Interactive 3D Graphics*, pp. 227–232, 2001.

[76] B. Lévy, "Constrained texture mapping for polygonal meshes," in *ACM SIGGRAPH 2001*, pp. 417–424, 2001.

[77] B. Lévy, "Dual domain extrapolation," in *ACM SIGGRAPH 2003*, 2003.

[78] B. Lévy and J.-L. Mallet, "Non-distorted texture mapping for sheared triangulated meshes," in *ACM SIGGRAPH '98*, pp. 343–352, 1998.

[79] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," in *ACM SIGGRAPH 2002*, pp. 362–371, 2002.

[80] W. C. Li, N. Ray, and B. Levy, "Automatic and interactive mesh to T-spline conversion," in *EG/ACM Symposium on Geometry Processing*, 2006.

[81] N. Litke, M. Droske, M. Rumpf, and P. Schröder, "An image processing approach to surface matching," in *Proceedings of the Symposium on Geometry Processing 2005*, 2005.

[82] F. Losasso, H. Hoppe, S. Schaefer, and J. Warren, "Smooth geometry images," in *Eurographics Symposium on Geometry Processing*, pp. 138–145, 2003.

[83] L. Lovasz and A. Schrijver, "On the null space of a Colin de Verdière matrix," *Annales de l'Institut Fournier*, vol. 40, no. 3, pp. 1017–1026, 1999.

[84] J. Maillot, H. Yahia, and A. Verroust, "Interactive texture mapping," in *ACM SIGGRAPH '93*, pp. 27–34, 1993.

[85] S. Marschner, B. Guenter, and S. Ragupathy, "Modeling and rendering for realistic facial animation," in *Proceedings of the Eurographics Workshop on Rendering Techniques*, pp. 231–242, 2000.

[86] J. McCartney, B. K. Hinds, and B. L. Seow, "The flattening of triangulated surfaces incorporating darts and gussets," *Computer-Aided Design*, vol. 31, no. 4, pp. 249–260, 1999.

[87] V. J. Milenkovic, "Rotational polygon containment and minimum enclosure," in *ACM Symposium on Computational Geometry*, 1998.

[88] J. Mitani and H. Suzuki, "Making papercraft toys from meshes using strip-based approximate unfolding," in *ACM SIGGRAPH 2004*, pp. 259–263, 2004.
[89] F. Morgan in *Riemann Geometry – A Beginner's Guide*, (A. K. Peters, ed.), (Massachusetts), Wellesley, 1998.
[90] X. Ni, M. Garland, and J. C. Hart, "Fair Morse functions for extracting the topological structure of a surface mesh," in *ACM SIGGRAPH 2004*, pp. 613–622, 2004.
[91] J. Pach and R. Wenger, "Embedding planar graphs at fixed vertex locations," in *Graph Drawing*, pp. 263–274, 1998.
[92] H.-K. Pederson, "Decorating implicit surfaces," in *ACM SIGGRAPH '95*, 1995.
[93] J. Peng, D. Kristjansson, and D. Zorin, "Interactive modeling of topologically complex geometric detail," in *ACM SIGGRAPH 2004*, 2004.
[94] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Experimental Mathematics*, vol. 2, no. 1, pp. 15–36, 1993.
[95] D. Piponi and G. Borshukov, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending," in *ACM SIGGRAPH 2000*, pp. 471–478, 2000.
[96] S. Porumbescu, B. Budge, L. Feng, and K. I. Joy, "Shell maps," in *ACM SIGGRAPH 2005*, 2005.
[97] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped textures," in *ACM SIGGRAPH 2000*, pp. 465–470, 2000.
[98] E. Praun and H. Hoppe, "Spherical parameterization and remeshing," in *ACM SIGGRAPH 2003*, pp. 340–350, 2003.
[99] E. Praun, W. Sweldens, and P. Schröder, "Consistent mesh parameterizations," in *ACM SIGGRAPH 2001*, pp. 179–184, 2001.
[100] B. Purnomo, J. Cohen, and S. Kumar, "Seamless texture atlases," in *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing 2004*, pp. 67–76, 2004.
[101] N. Ray and B. Lévy, "Hierarchical least squares conformal maps," in *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pp. 263–270, 2003.
[102] N. Ray, W.-C. Li, B. Lévy, A. Sheffer, and P. Alliez, "Periodic global parameterization," *ACM Transactions on Graphic*, vol. 25, no. 3, 2006.
[103] A. Rockwood and H. Park, "Interactive design of smooth genus N objects using multiperiodic functions and applications," *International Journal of Shape Modeling*, vol. 5, pp. 135–157, 1999.
[104] B. Rodin and D. Sullivan, "The convergence of circle packings to the Riemann mapping," *Journal of Differential Geometry*, vol. 26, no. 2, pp. 349–360, 1987.
[105] S. Saba, I. Yavneh, C. Gotsman, and A. Sheffer, "Practical spherical embedding of manifold triangle meshes," in *Proceedings of Shape Modeling International*, 2005.
[106] P. Sander, S. Gortler, J. Snyder, and H. Hoppe, "Signal-specialized parametrization," in *Eurographics Workshop on Rendering*, pp. 87–100, 2002.
[107] P. Sander, J. Snyder, S. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *ACM SIGGRAPH 2001*, pp. 409–416, 2001.

[108] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe, "Multi-chart geometry images," in *ACM Symposium on Geometry Processing 2003*, 2003.
[109] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe, "Inter-surface mapping," in *ACM SIGGRAPH 2004*, 2004.
[110] A. Shamir, "A formulation of boundary mesh segmentation," in *International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 82–89, 2004.
[111] A. Shapiro and A. Tal, "Polyhedron realization for shape transformation," *The Visual Computer*, vol. 14, no. 8, pp. 429–444, 1998.
[112] A. Sheffer, "Spanning tree seams for reducing parameterization distortion of triangulated surfaces," in *Shape Modelling International*, pp. 61–66, 2002.
[113] A. Sheffer and E. de Sturler, "Surface parameterization for meshing by triangulation flattening," *Proceedings of 9th International Meshing Roundtable*, pp. 161–172, 2000.
[114] A. Sheffer and E. de Sturler, "Parameterization of faceted surfaces for meshing using angle based flattening," *Engineering with Computers*, vol. 17, no. 3, pp. 326–337, 2001.
[115] A. Sheffer and E. de Sturler, "Smoothing an overlay grid to minimize linear distortion in texture mapping," *ACM Transaction on Graphics*, vol. 21, no. 4, pp. 874–890, 2002.
[116] A. Sheffer, C. Gotsman, and N. Dyn, "Robust spherical parameterization of triangular meshes," in *4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics*, pp. 94–99, 2003.
[117] A. Sheffer and J. Hart, "Seamster: Inconspicuous low-distortion texture seam layout," in *IEEE Visualization*, pp. 291–298, 2002.
[118] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomyakov, "ABF++: Fast and robust angle based flattening," *ACM Transactions on Graphics*, vol. 24, no. 2, pp. 311–330, 2005.
[119] C. Soler, M.-P. Cani, and A. Angelidis, "Hierarchical pattern mapping," *ACM SIGGRAPH 2002*, pp. 673–680, 2002.
[120] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, "Bounded-distortion piecewise mesh parametrization," in *IEEE Visualization*, pp. 355–362, 2002.
[121] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössland, and H.-P. Seidel, "Laplacian surface editing," in *Proceedings of Eurographics/ACM Symposium on Geometry Processing*, pp. 179–188, 2004.
[122] R. Sumner and J. Popović, "Deformation transfer for triangle meshes," in *ACM SIGGRAPH 2004*, pp. 399–405, 2004.
[123] V. Surazhky and C. Gotsman, "Explicit surface remeshing," in *ACM/Eurographics Symposium on Geometry Processing*, 2003.
[124] M. Tarini, K. Hormann, P. Cignoni, and C. Montani, "Poly-cube maps," in *ACM SIGGRAPH 2004*, pp. 853–860, 2004.
[125] G. Tewari, J. Snyder, P. Sander, S. Gortler, and H. Hoppe, "Signal-specialized parameterization for piecewise linear reconstruction," in *Eurographics Symposium on Geometry Processing*, pp. 57–66, 2004.

[126] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun, "Designing quadrangulations with discrete harmonic forms," in *Symposium on Geometry Processing*, 2006.
[127] G. Turk, "Texture synthesis on surfaces," in *ACM SIGGRAPH 2001*, pp. 347–354, 2001.
[128] W. T. Tutte, "How to draw a graph," *London Mathematical Society 1963*, vol. 13, pp. 743–768, 1963.
[129] L.-Y. Wei and M. Levoy, "Texture synthesis over arbitrary manifold surfaces," in *ACM SIGGRAPH 2001*, 2001.
[130] Wikipedia, Normal Mapping. http://en.wikipedia.org/wiki/Normal_mapping.
[131] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin, "Texture and shape synthesis on surfaces," in *Eurographics Workshop on Rendering*, 2001.
[132] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, "A fast and simple stretch-minimizing mesh parameterization," in *Shape Modeling International*, 2004.
[133] R. Zayer, C. Rossl, and H.-P. Seidel, "Variations on angle based flattening," in *Proceedings of Multiresolution in Geometric Modelling*, pp. 285–296, 2003.
[134] R. Zayer, C. Rössl, and H.-P. Seidel, "Discrete tensorial quasi-harmonic maps," in *Proceedings of Shape Modeling and Applications*, pp. 276–285, 2005.
[135] R. Zayer, C. Rössl, and H.-P. Seidel, "Setting the boundary free: A composite approach to surface parameterization," in *Symposium on Geometry Processing*, pp. 91–100, 2005.
[136] R. Zayer, C. Rössl, and H.-P. Seidel, "Curvilinear spherical parameterization," in *Proceedings of Shape Modeling and Applications*, pp. 57–64, 2006.
[137] E. Zhang, Mischaikow, and G. Turk, "Feature-based surface parameterization and texture mapping," *ACM Transaction on Graphics*, vol. 24, no. 1, pp. 1–27, 2005.
[138] K. Zhou, J. Snyder, B. Guo, and H.-Y. Shum, "Iso-charts: Stretch-driven mesh parameterization using spectral analysis," in *Eurographics Symposium on Geometry Processing*, pp. 47–56, 2004.
[139] K. Zhou, X. Wang, Y. Tong, M. Desbrun, B. Guo, and H.-Y. Shum, "Texture Montage: Seamless texturing of arbitrary surfaces from multiple images," in *ACM SIGGRAPH 2005*, 2005.
[140] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture mapping using surface flattening via multi-dimensional scaling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 2, pp. 198–207, 2002.