Vis Comput (2009) 25: 589–597
DOI 10.1007/s00371-009-0334-4

ORIGINAL ARTICLE

Adaptive smooth surface fitting with manifolds

Cindy Grimm · Tao Ju · Ly Phan · John Hughes

Published online: 5 March 2009

Abstract We present a smooth, everywhere C^k, analytic surface representation for closed surfaces of arbitrary topology. We demonstrate fitting this representation to meshes of varying resolutions and sampling quality. The fitting process is adaptive and provides controls for both the average and the maximum allowable error. The representation is suitable for applications which require consistent parameterizations across different surfaces.

Keywords Manifolds · Surface fitting · Analytic surface

1 Motivation

We present a smooth, everywhere C^k, analytic surface representation for closed surfaces of arbitrary topology. In order for a representation to be useful, we need to be able to build surfaces from data. In this paper, we show how to fit our representation to an input mesh of the same topology, taking advantage of the adaptive nature of the representation to better control both average and maximum fit levels. Most fitting approaches typically minimize just the average error; this can lead to areas which are inadequately represented, even though the average is still low. We address this by allowing the user to specify both a desired average and maximum error.

1.1 Overview

In our representation the surface is defined as an embedding of a global domain of the desired topology, such as a sphere or torus. The embedding is created by locally describing what the surface should look like and blending the results together. Unlike splines, the continuity of the surface is maintained by the blending process, not constraints between the local descriptions. This means that these local surface pieces can be added, changed, or deleted without needing to re-establish continuity constraints.

More specifically, we define a general method for defining local, planar parameterizations on each of the basic global domain topologies (sphere, torus, or n-holed torus). This parameterization is a C^∞, invertible map from part of the global domain to a disk in the plane. The embedded surface is created by writing an embedding function for each of these local parameterizations, then blending them together based on how they overlap in the global domain.

Our surface representation naturally supports additional data sets with different resolution requirements by simply defining new local data descriptions on the global domain.

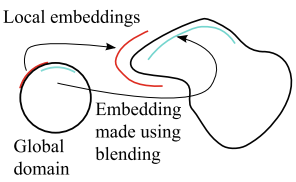

Fig. 1 Defining a surface using our representation. Surface is an embedding of a global domain (in this case a circle). The embedding is defined by several local embeddings (two shown) which are blended together

C. Grimm (✉) · T. Ju · L. Phan
Washington University in St. Louis, St. Louis, USA
e-mail: cmg@cs.wustl.edu

T. Ju
e-mail: taoju@cse.wustl.edu

L. Phan
e-mail: faanly@cs.wustl.edu

J. Hughes
Computer Science Department, Brown University,
Box 1910, Providence, RI 02912, USA
e-mail: jfh@cs.brown.edu

The correspondence between the additional data and the geometry is maintained through the global domain.

To build our surface representation from a data set we use a modified least-squares-based approach. The input data is a manifold mesh of the desired topology. The fitting process has two steps: (1) Build a mapping between the global domain and the mesh. (2) Adaptively cover the domain (and mesh) with local parameterizations, each of which is fit to the corresponding part of the input mesh. This fitting process is entirely local and does not require any geometric constraints between overlapping parameterizations. We control the quality of the fit through the size of the local parameterizations.

Contributions Building a C^k analytic surface model from a mesh. The surface is guaranteed to have the same topology as the input mesh. The model and fitting process have the following properties:

- Adaptive surface reconstruction with explicit control over both average and maximum allowable error.
- Explicit control over the region of influence of an individual chart.
- Parameterization adjustment without fold-over.
- Embedding of an n-holed torus mesh into a $4n$-sided polygon in the Poincaré disk.

The paper is organized as follows: We first define the surface representation in Sect. 3. In Sect. 4 we show how to adapt a least-squares approach to adaptively fit our representation to a data set. Section 4.4 defines how to build blend functions with specific supports. We close with results.

2 Previous work

A full review of all surface representation techniques is beyond the scope of this paper; we focus on methods that are analytical and parametric. We break these methods into two groups, those that use a global domain and those that define the surface domain through local parameterizations.

Global domains One approach is to build an affine parameterization from the entire input mesh. Unless the mesh is planar or toroidal, this requires punching a small number of holes in the mesh before creating the affine mapping. The advantage of this approach is that the domain is very simple and easy to build a surface using, for example, T-splines [1–3] or triangular splines [4]. A similar approach uses a polycube [5] as a base domain; again, the domain has a small number of discontinuities. For all of these approaches, some form of geometric patching is added to fix the surface around the discontinuities. The T-splines provided limited adaptive refinement by adding more control points.

A second approach is to use a smooth global domain, such as the sphere or tiled hyperbolic plane, and define a general technique for building local parameterizations over that domain. This is the approach taken here. Earlier work used this approach for the sphere, tiled plane, and tiled hyberbolic plane, but with a fixed set of a small number of parameterizations [6]. In [7] this was extended to creating arbitrary local parameterizations, but only for the sphere.

Other work [8–11] has used the tiled hyperbolic disk for n-holed surfaces by mapping an input mesh to the disk, cutting where necessary to produce the tiling. None of these approaches produced satisfactory analytic surfaces from the tiling, much less fitted surfaces.

Local domains B-splines are a well-established approach, but because they are planar, other surface topologies must be built using multiple patches. Early work [12] relied on hand-segmentation to divide the surface into patches, then fit each patch through a relaxation method. Guaranteeing continuity, particularly greater than C^1, between patches while fitting is still an open area of research [13], although commercial software exists [14] that is fairly reliable.

Subdivision surfaces [15–19] remove the need for explicitly maintaining continuity, but while regular areas of the subdivision mesh can be approximated by splines, the surface is not everywhere analytic.

Several manifold-based approaches exist that build the domain by defining local charts and their overlaps [20–22]. The topology of the domain is defined by a mesh. Charts are built for some (or all) of the elements of the mesh, with the chart overlaps defined by the element adjacencies. The domain is defined by explicitly defining transition functions that glue the charts together.

None of the above approaches allow for control over the maximum error, and only the T-splines approach has support for adaptively adjusting the fitted surface.

3 Surface representation

The surface representation consists of three parts. The first is an explicit representation of the global domain D for each genus (sphere, torus, or n-holed tori, Fig. 2) [23]. The second part is a general mechanism for creating a local parameterization on the domain. The local parameterization is broken into two steps; the first step maps from the domain D to the plane, the second step adjusts the mapping with an affine transformation (see Fig. 3). The third part is the building of the surface (or other function) itself. The domain D is embedded by defining blend functions and embeddings for each of the individual local parameterizations. These individual embeddings are blended together (using the overlap information and blend functions) to produce the final surface. This is analogous to the spline approach, except we are blending functions instead of control points.

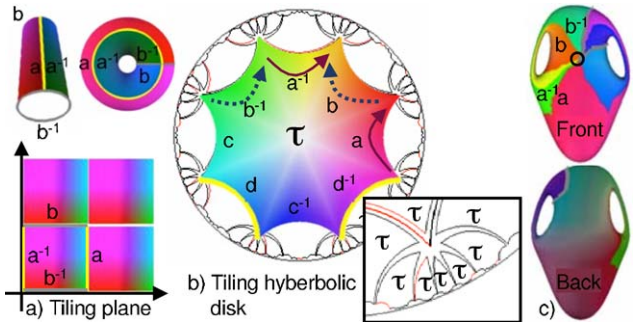

Fig. 2 (**a**) Tiling the plane to make a torus. Top: Gluing the square together along a and a^{-1}, then b and b^{-1}, to make a torus. (**b**) Tiling the hyberbolic disk with an 8-sided polygon. (**c**) Showing a single copy of the polygon τ wrapped onto the vase mesh

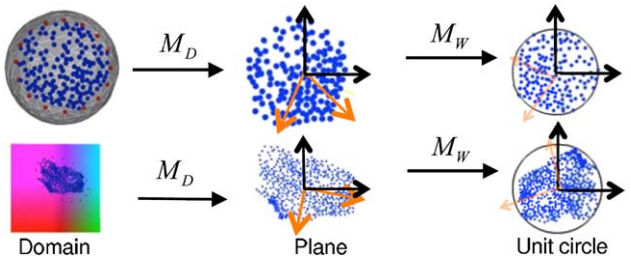

Fig. 3 Constructing chart functions for the spherical (*top*) and tori (*bottom*) cases. M_D maps from the domain to the plane, M_W rotates and scales the area of interest (*blue dots*) to fit in the *circle*. *Yellow* arrows are eigen vectors, *red dots* are the chart boundary mapped back to the domain

The domains We use a unit sphere for surfaces that are topologically spheres. For n-holed tori, we tile the hyperbolic disk with a $4n$-sided polygon (Fig. 2b). For $n = 1$, this simplifies to the tiled Euclidean plane (Fig. 2a).

The local parameterizations To create a local parameterization, we define a mapping function α_c that takes a portion $D_c \subset D$ of the domain to a region c in the plane. To simplify matters, in this paper we always use a unit disk centered at the origin for all of our regions c, although c can actually be any shape. D_c, α_c, and c are collectively referred to as being a *chart*.

The local parameterization, depicted schematically in Fig. 3, consists of a domain D-dependent map M_D from D to the plane, followed by a rotation, translation, and scale M_W, i.e., $\alpha_c = M_W \circ M_D$. Both M_D and M_W must be invertible over the area of interest. The map M_D is used to select where the center of the chart is located, while M_W adjusts the orientation and size of the chart's domain, D_c, which is defined as $D_c = (M_D^1 \circ M_W^{-1})(c)$.

For the sphere, M_D is a rotation of the sphere, followed by a stereographic projection [7]. For the torus, M_D is a translation, re-centering the tiled plane at the given point. For n-holed tori, M_D is a Linear Fractional Transform. Let $p = r(\cos\theta + i \sin\theta)$ be the desired center of the chart. Then the centering transform is:

$$M_D = \begin{bmatrix} \cos -\theta + i \sin -\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -r \\ -r & 1 \end{bmatrix}. \quad (1)$$

Note that α_c is valid for a substantial portion of D; we restrict the mapping to $\alpha_c(c)$ when embedding to control how much of D (and hence the surface) the chart covers.

Transition functions between charts i and j are built by composition $\alpha_i \circ \alpha_j^{-1}$. Since the chart functions are C^∞ and invertible, the transition functions are as well, yielding a C^∞ manifold structure.

3.1 Defining a function on the domain

For each chart, we create an embedding function E_c and a blend function B_c. The embedding function describes what the surface should look like locally, while the blend function defines the influence of the chart's embedding function.

The embedding functions E_c are polynomials that map c to $\mathbb{R}^3$; we use a polynomial degree of four, which is sufficient to provide flex in the surface. The blend functions B_c are C^k smooth "bumps" which are one at the origin and zero, along with the first k derivatives, at the boundary (see Sect. 4.4).

The global function is a blended combination of the local functions. Given a point $p \in D$,

$$E(p) = \frac{\sum_c B_c(\alpha_c(p)) E_c(\alpha_c(p))}{\sum_c B_c(\alpha_c(p))} \quad (2)$$

where we define $B_c(\alpha_c(p))$ to be zero if $p \notin D_c$. We ensure that the denominator is nonzero by ensuring that $\sum_c B_c$ is nonzero everywhere. The continuity of E_c depends on the minimum continuity of its constituent parts. In this paper the α_c and E_c functions are both C^∞; the blend functions are C^2. Unlike splines, changing the continuity of the blend function does not dramatically change the *visual* appearance of the surface because it only slightly alters the blend function's shape.

4 Fitting to a mesh

In this section we describe how to fit our representation to a manifold mesh of the desired topology. The fitting proceeds in two steps (see Fig. 4):

- Let D be the domain with the same topology as the input mesh. Create a 1–1, onto mapping between the input mesh and D by embedding the mesh into D.
- Create an initial set of N_c charts that cover D by partitioning the mesh. Each chart covers its partition and a bit more. Adaptively add charts until the fit error is below the user-specified average and maximum error.

The user provides four parameters that control the chart placement. The first two are the desired average E_a and maximum E_m allowable point error. The second two are the desired average En_a and maximum En_m allowable normal variation. We now discuss these steps in detail.

4.1 Embedding the input mesh

The goal of the embedding step is to distribute the vertices in the domain, respecting the local neighborhood connectivity. Like most parameterization algorithms, the relative local geometry placement in the domain should be similar to the geometry in 3D (relative edge lengths and angles).

For the sphere, we use Saba's approach [24], although any other approach would work as well. For the toroidal case, we start with Erickson's approach [25, 26], which creates initial $2n$ loops (two for each hole) through a vertex V_s. We then change the labeling of these loops to match our $4n$-sided polygon labeling.

First cut the mesh open along the loops, starting at v_s and tracing along the edges in the loops. This tracing process will visit each loop twice, the second time in the reverse direction. Label each part of the boundary with its corresponding loop $a, b, \ldots$, labeling the second occurrence as $a^{-1}, b^{-1}, \ldots$. For our global domain, we need the boundary labeled as $aba^{-1}b^{-1}, \ldots, yzy^{-1}z^{-1}$. Slice the disk and reglue it together (see Fig. 5) until the loops are in the correct order.

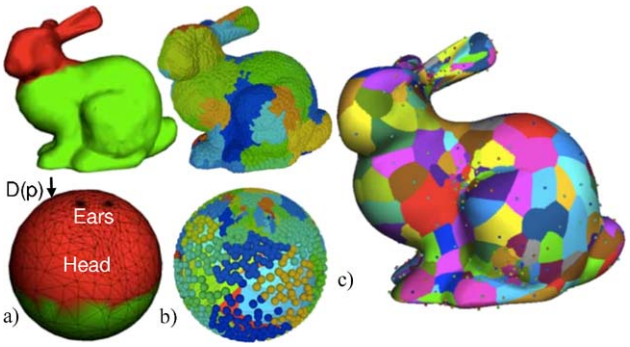

Fig. 4 Fitting process. (**a**) Embed the mesh in the domain. (**b**) Partition the mesh (and the domain) into charts. (**c**) Fit each chart to its corresponding part of the mesh

Slicing proceeds by finding a pair of labels of the form $abZa^{-1}Rb^{-1}L$, where Z, R, and L are (possibly empty) pieces of the boundary. Cut along a path from the end of a to the end of b^{-1} and glue the removed piece back together along a. Next, cut from the end of a to the beginning of a^{-1} and glue the disk back together along b. The result is the correct labeling for one hole; repeat until all holes are correctly labeled. This disk is embedded in the polygon by matching corners and edges.

Guaranteeing no fold-overs When iteratively adjusting positions, we constrain the movement of each vertex to a safe region. For the sphere, this safe region is built in the following way. For each triangle adjacent to the vertex, build three planes. Each plane passes through the origin and the midpoints of two adjacent triangle edges. Define the plane normal so that the vertex is on the positive side of the plane. The safe region is the union of all of these half-spaces. For the toroidal case, the planes are all perpendicular and can be replaced by lines [27].

4.2 Placing charts

The goal of the chart placement algorithm is to cover the mesh with as few charts as possible while enforcing that each chart be adequately fit to its corresponding part of the mesh. The charts must also overlap so that there is a smooth transition from one chart to the next.

The algorithm proceeds by placing an initial set of charts, then adaptively adding charts where the fit is not good enough. To ensure coverage we partition the mesh into regions, one for each chart. To ensure overlap, we expand each region by one ring of faces. The chart is responsible for covering this expanded region.

A chart is specified by its location (M_D) and its scale and orientation (M_W). We place each chart so that its center is at the center of the region, then adjust M_W until the chart just covers the assigned region.

The algorithm has three components: How to place the region centers, how to build the regions, and how to calculate M_D and M_W. We use Lloyd's [28] algorithm, which evenly distributes points on the mesh using Voronoi relaxation, to determine the centers of the regions. The regions

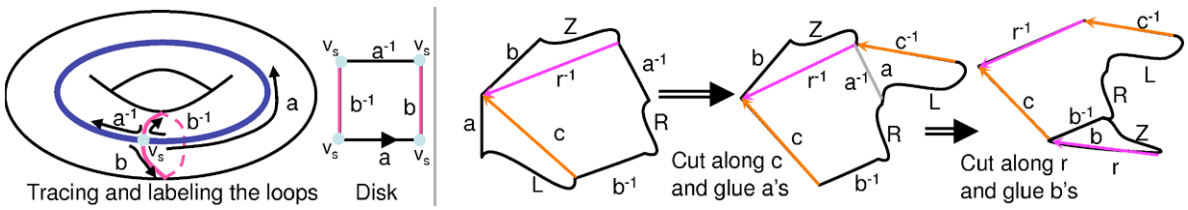

Fig. 5 *Left*: Tracing along the loops to cut the mesh into a disk. Cut along a, then b, then a^{-1}, then b^{-1}. *Right*: Fixing an existing disk so that one hole ($aba^{-1}a^{-1}$) is correctly labeled. Repeat on the remaining LRZ boundary

themselves are then determined by doing a Voronoi tessellation in the *domain* D. Finally, we use an optimization procedure to "shrink wrap" each chart around its region.

Placement We use binary search on the number of centers, aiming for 70% of the mesh being adequately fit. These centers are uniformly distributed on the mesh using Lloyd's algorithm. We then iteratively add more centers both between the centers of adjacent, poorly fitted charts, and at the edge of isolated, poorly fit ones. Only these newly added centers, and the centers of the poorly fitted charts, are allowed to move in subsequent Lloyd relaxation steps. This process stops when there are no more poorly fitted charts or when there is no place to add new centers.

Making regions We could use the Voronoi regions produced by previous step as our regions. This often, however, results in crescent-shaped regions in the domain in places like the base of the bunny's ear. Therefore, we do a Voronoi tessellation using geodesic distance in *domain* space and using the centers (mapped to D) that were found in the previous step. We then add one more ring of faces around each Voronoi region.

Making charts Let $\mathbb{D}$ be the function that maps the mesh to the domain D, and R_c be the region centers found by the Lloyd's algorithm. Set M_D so that $\alpha_c(\mathbb{D}(R_c))$ is the origin. Let $\mathbb{D}(\{P\}_c)$ be the set of domain points found by mapping the region's vertices and boundary-edge midpoints to the domain. We next solve for the affine map M_W that "shrink wraps" the chart's boundary around $\mathbb{D}(\{P\}_c)$.

First, map the points to the plane using M_D ($\{p_c\} = M_{D_c}(\{P\}_c)$). Next, use Principal Components Analysis on the $\{p_c\}$ points to find an initial scale and orientation. Set M_W to rotate the first eigen vector to the x axis and set the scales to be the eigen values. The initial translation is set to zero because the chart is already roughly centered. Finally, optimize over the rotation and translation in order to minimize the area of $M_W^{-1}(c)$. At each iteration, use binary search to find the smallest x and y scale such that $M_W(p_c) \subset c$. This avoids the problem of trying to include that constraint in the optimization term.

4.3 Chart fitting

The embed function E is a K-degree polynomial. We use as a starting point the standard approach of solving for the coefficients using a least-squares approach, where each row represents a point constraint. We modify this to allow weighting for each row and to allow face normal constraints. A face normal constraint n is added by constraining the dot product of the face normal and the derivative at the face's centroid to be zero. To set the length of n, fit E using only the point constraints. Set n's length to be one over the (unnormalized) length of E's surface normal at that point.

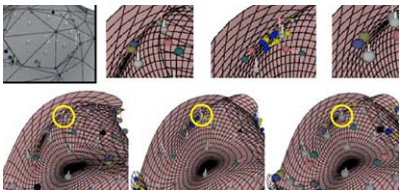

Fig. 6 *Left*: Fitting to just the points. *Middle*: Fitting to the points and normals with adjusted weights. *Right*: Nonlinear optimization. *Circled area* shows a normal which is initially flipped

For our point constraints, we take all of the vertices in the region, plus the midpoints of the edges that lie on the boundary of the region. This helps to counteract the fact that interior points have more influence. The weights for the point constraints are initially set to $1/N_p$, where N_p is the number of point constraints.

For our normal constraints, we take all of the face normals in the region, plus the normals at the midpoints of the edges that lie on the boundary. The initial weights for the normal constraints are E_a/N_n, where E_a is the average allowed point error, and N_n is the number of normal constraints.

This approach solves for the solution with the minimum root mean squared error (RMS). Unfortunately, it often does so by increasing the error to the boundary points, resulting in a surface which has a good RMS but possibly high maximum error (see Fig. 6). We therefore allow average error to increase if it means decreasing the error at any points with large error. We describe two optimization approaches that accomplish this, one of which is relatively fast and involves iteratively adjusting the weights in the matrix, the second which is slower and is a general optimization.

Iterative weighting The iteration proceeds as follows. At each step, the weights of each constraint are adjusted, and the surface is refit. We cap the number of iterations at twenty because, after experimentation with several data sets, we never saw improvement after this. Let $l = \|E(u, v) - P\| > E_m$ and $\delta_p = 2/N_p$. Then the weight is changed by

$$\delta w = \begin{cases} \delta_p + \delta_p \frac{l - E_m}{E_m} & l \geq E_m, \\ \delta_p \frac{l - E_a}{E_m - E_a} & E_m > l > E_a. \end{cases} \tag{3}$$

We update the normal constraint weights in a similar fashion, letting δ_p be $2E_m/N_n$. Additionally, if the dot product is *negative*, we add additional point constraints to unfold the surface there. For face normals, we add three points surrounding the face centroid. For edge normals, we add four points, two on the edge surrounding the mid point, the other two part way into the faces on either side (see Fig. 6, middle).

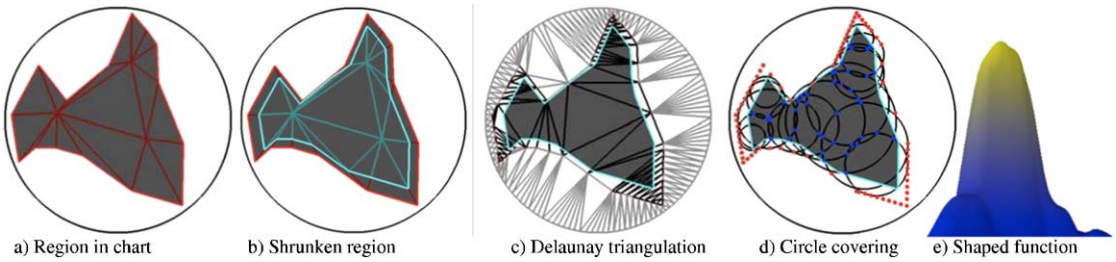

Fig. 7 Building a Shaped Blend Function. *Red boundary*: The expanded region of the mesh that the chart covers, mapped to the domain of the chart. *Cyan boundary*: The region shrunk by moving 1/4 in on each edge that touches the boundary. In (**c**) we sample the chart and region boundary and create a Delaunay triangulation of the *original* region which completely encloses the shrunken region. (**d**) Each Delaunay triangle generates a circle. A subset of these circles, who's union covers the shrunken region, are selected. (**e**) The resulting blend function

General optimization The above approach works in most cases but can have trouble unfolding edges. In these cases we apply a general solver with these terms:

Average terms: For the point constraints, we calculate the average error ε_a and weight it by the average desired error, yielding $\varepsilon_a/(2E_a)$. For the normal constraints, we calculate the average dot product error εn_a and then take $(1-\varepsilon n_a)/2$. We weight both average terms by $1/2$ to emphasize the maximum error.

Maximum terms: For the point constraints, we add $1+\min(1, l-E_m)/(10E_m)$, where l is the distance to the desired point, for every point where $l > E_m$. For the normal constraints, we add $1+(1-d)/2$, where d is the dot product, for every point where $d < En_m$.

4.4 Shaped blend functions

The partition process (Sect. 4.2) tries to assign regions to charts so that, when mapped to the chart, the regions are as close to the shape of the chart as possible. We can further control the region of influence of the chart by using *shaped* blend functions, matching the function's support to the shape of the region.

To ensure that our manifold embeddings are valid, we must ensure that the blend function supports overlap—every point on the mesh is covered by one (or more) nonzero blend functions. We guarantee this by ensuring that each shaped blend function covers its part of the mesh plus (at a minimum) approximately $3/4$ of the ring around the region. This puts the total overlap between two adjacent charts at roughly $11/2$ faces.

Definition Our shaped blend functions consist of the sum of individual circular "bumps":

$$\hat{B}_c(s,t) = \sum_i w_i \beta\left(\frac{(s-s_i)^2+(t-t_i)^2}{r_i^2}\right), \tag{4}$$

$$\beta(d) = \begin{cases} 0 & d > 1, \\ 1-10d^3+15d^4-6d^5 & d \le 1, \end{cases} \tag{5}$$

where (s_i, t_i, r_i) is a circle centered at (s_i, t_i) with radius r_i, and w_i is a constant that scales the height of the bump. All of the circles must be contained within the domain of the chart. β is a polynomial which is 1 when $d = 0$, 0 when $d = 1$, and the first and second derivatives at 0 and 1 are both 0. This provides C^2 continuity. Continuity C^k (including C^∞) can be achieved by changing β [21].

Construction Refer to Fig. 7. Sample both the boundary of the region and the chart to create a Delaunay triangulation. Take only the triangles that cover the interior of the region. Each of these triangles represents a possible circle we can use (the circle that passes through the triangle's vertices). Clearly, the union of all of these circles covers the region because each circle covers its corresponding triangle.

Rather than use all of the circles, we take a subset of them. Take the boundary of the region and shrink it slightly by moving it $1/4$ in. Greedily choose circles which cover the most of the remaining uncovered part of the boundary, until all of the shrunken boundary is covered. To ensure that there are no gaps in the interior of the region, all circle–circle intersections that lie in the shrunken region must also be covered by some *other* circle. This guarantees that the boundary of the union of the circles lies outside of the shrunken region.

To guarantee that the circles do not extend outside of the circle's domain, first increase the size of the chart slightly so that the region boundary is at least 0.05 from the chart's boundary. Sample the boundary so that the samples are closer to each other than they are to the chart's edge. This prevents the circles from expanding past the region boundary and crossing the edge.

Weights Once we have our circles, we pick a weight w_i for each one so that the shape of Eq. 5 is as "bump-like" as possible and so that the derivatives of B_c are as small as possible. We accomplish this using a constrained least-squares approach, solving for positive weights that minimize the derivatives while making $B_c(0,0) = 1$.

4.5 Additional steps

We can optionally adjust the parameterization to reduce the error in the tangent plane direction. We fit the surface, then for each data point, find the closest point on the fitted surface. This defines the desired parameter point in the domain for that data point. We then iteratively move the domain points towards their desired locations while preventing fold-overs (Sect. 4.1).

To reduce the number of shaped blend functions, we check to see if the chart can adequately fit the region of D it covers; if so, we replace the shaped blend function with a single circle which exactly covers the chart. For the fit check, we use one point per vertex covered by the chart, and one point per face. We additionally add 16 point constraints for the boundary by mapping evenly spaced samples on the chart's boundary back to the mesh.

5 Results and remarks

Figure 10 illustrates the effect of changing both the desired fit E_a and the mesh resolution. Figure 11 compares our method with GeoMagic's results. Figures 8 and 9 show one- and two-holed tori examples, respectively. Note on color: Gray to white is below E_a, blue-yellow is between E_a and E_m, and red is greater than E_m. The input parameters and output results for all figures are summarized in Table 1. Max constraints are not met when it is not possible to fit a polynomial to the face plus the one ring neighborhood. The accompanying video shows the places were the maximum constraints were not met. All normals used in rendering are computed analytically.

Running times are on the order of a few minutes to a few hours, depending on the density of the mesh, the desired error, and the quality of the parameterization. Polynomials are

Table 1 N_v is the number of vertices. E_a is the average point error, as a percentage of the diagonal of the bounding box. For all surfaces, $E_m = 3E_a$, $En_a = 0.9$, $En_m = 0.6$, polynomial degree is 4. $En_a = 0.9$, except bones, which was $En_a = 0.6$, $En_m = 0.3$. Charts is the number of charts, shp is the percentage of shaped blend functions. The point averages are given as a percentage of the bounding box diagonal. The max values are the number of constraints which were not met

Data		Inputs		Outputs		
Data	N_v	E_a	Charts	Shp	Pt	Norm
Bones	49701	3.3	3009	0	1.2, 20	0.63, 112
Bunny	1502	1	477	56	0.67, 0	0.97, 14
	1502	0.1	2940	86	0.16, 112	0.99, 2
	15002	1	5945	28	0.47, 0	0.97, 333
	15002	0.1	10075	59	0.12, 17	0.99, 10
Garg	10002	1	4513	50	0.3, 0	0.95, 37
Vase	1476	1	812	100	0.4, 0	0.92, 18
Rocker	10044	0.2	5066	48	0.2, 44	0.99, 2
Buste	5002	1	2719	74	0.3, 0	0.97, 9
Bimba	4502	0.2	3731	64	0.2, 4	0.97, 21
	7502	5	9206	64	0.8, 0	0.86, 1458

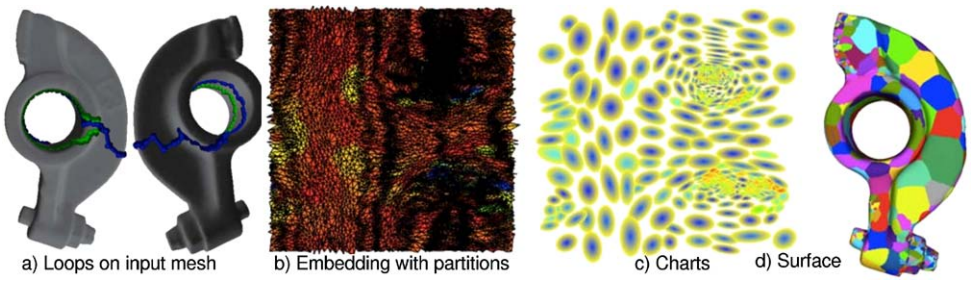

Fig. 8 (**a**) Initial cut loops on input mesh. (**b**) Embedding of mesh into torus domain, colored by partition. (**c**) Charts on domain, shrunk to 1/4 size. (**d**) Reconstructed surface, colored by chart center

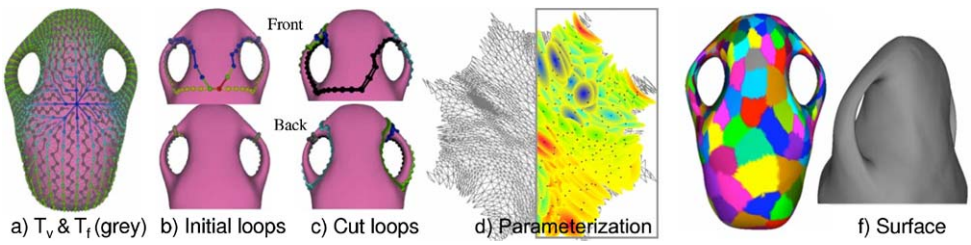

Fig. 9 2-holed tori. (**a**) Initial loops (**b**) Fixed loops. (**c**) The mesh after cutting and relaxation (70 iterations). Chart coverage, *right*: *blue dots* are chart centers, *colors* indicate chart order. (**d**) The reconstructed surface, colored by chart

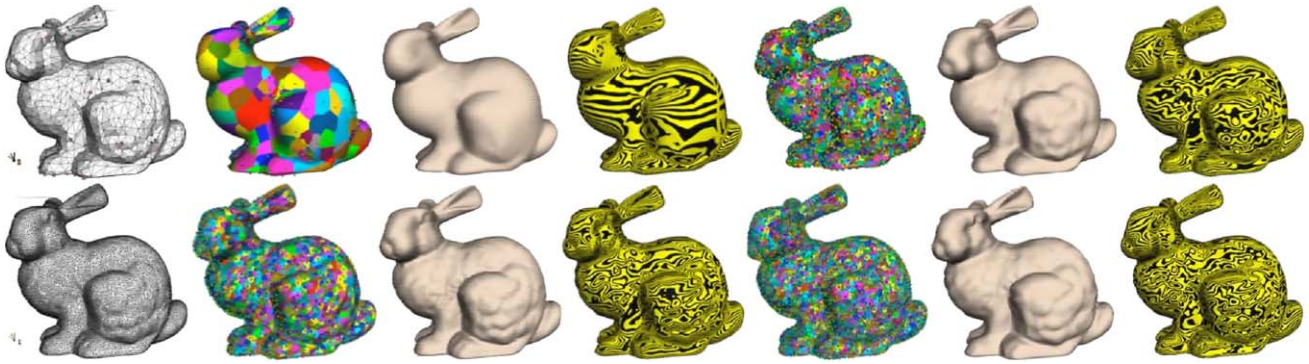

Fig. 10 *Top*: low-res bunny. *Bottom*: hi-res bunny. *Left*: 1% error. *Right*: 0.1% error. Shown is the original mesh, the output surface colored by chart and with chart centers, and reflection and shaded renderings

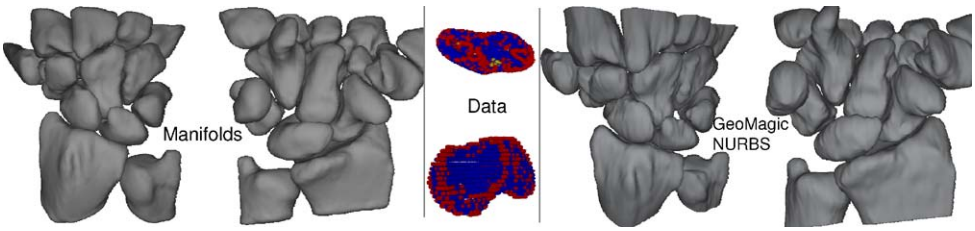

Fig. 11 The bones of the hand constructed from CT scans. *Left*: Our reconstruction, average absolute error 0.16. *Middle*: Example data showing contour structure. *Right*: NURBS surfaces constructed using commercial software (Geomagic), average error 0.13

less sensitive than spline patches to "skew" in the parameterization but can still fail even when the geometry is relatively flat. Adjusting the parameterization (Sect. 4.5) greatly reduces the tangential error, and hence the number of needed charts, but is fairly expensive computationally (4988 to 3960 for the rocker arm). Similarly, weighted fitting (Sect. 4.3) is expensive but can significantly drop the number of charts needed for the same level of fit (866 to 477 for the low-res bunny, 3960 to 1825 for the rocker arm).

We primarily envision using this representation where subsequent processing depends on smooth surfaces [29] or for consistently parameterizing similar surfaces. For surfaces that have sharp features, such as CAD/CAM objects, it is possible to define an embedding function which explicitly models the tangent discontinuity, using, for example, a spline surface with collapsed knots.

In conclusion, we have presented a technique for adaptively placing charts over a surface based on user-specified average and maximum errors. The resulting surface is analytic and locally parameterizable. The technique is robust to different sampling patterns (contours and poorly shaped triangles).

Acknowledgements This work was funded in part by NSF grants CCF 0702662 and CCF 0429856.

References

1. He, Y., Wang, K., Wang, H., Gu, X., Qin, H.: Manifold T-spline. In: GMP '06, pp. 409–422 (2006)
2. Zheng, J., Wang, Y., Seah, H.S.: Adaptive T-spline surface fitting to z-map models. In: GRAPHITE '05, pp. 405–411 (2005)
3. Li, W.-C., Ray, N., Lévy, B.: Automatic and interactive mesh to T-spline conversion. In: SGP 06: Symposium on Geometry Processing, pp. 191–200, Eurographics Association (2006)
4. Gu, X., He, Y., Qin, H.: Manifold splines. In: SPM '05, pp. 27–38 (2005)
5. Wang, H., He, Y., Li, X., Gu, X., Qin, H.: Polycube splines. In: SPM '07, pp. 241–251 (2007)
6. Grimm, C., Hughes, J.: Modeling surfaces of arbitrary topology using manifolds. Comput. Graph. **29** (1995)
7. Grimm, C.: Spherical manifolds for adaptive resolution surface modeling. In: Graphite, November 2005
8. Ferguson, H., Rockwood, A.: Multiperiodic functions for surface design. Comput. Aided Geom. Des. **10**, 315–328 (1993)
9. Rockwood, A., Park, H.: Interactive design of smooth genus *n* objects using multiperiodic functions and applications. Int. J. Shape Model. **5**, 135–157 (1999)
10. Wallner, J., Pottmann, H.: Spline orbifolds. In: Curves and Surfaces with Applications in CAGD, pp. 445–464 (1997)
11. Gu, X., Yau, S.-T.: Global conformal surface parameterization. In: SGP '03, pp. 127–137 (2003)
12. Krishnamurthy, V., Levoy, M.: Fitting smooth surfaces to dense polygon meshes. In: SIGGRAPH '96, pp. 313–324 (1996)
13. Shi, X., Wang, T., Wu, P., Liu, F.: Reconstruction of convergent G1 smooth B-spline surfaces. Comput. Aided Geom. Des. **21**(9), 893–913 (2004)
14. Geo: Geomagic commercial software

15. Lee, A., Moreton, H., Hoppe, H.: Displaced subdivision surfaces. In: SIGGRAPH '00, pp. 85–94. ACM Press/Addison-Wesley, New York (2000)
16. Litke, N., Levin, A., Schröder, P.: Fitting subdivision surfaces. In: VIS '01, pp. 319–324. IEEE Computer Society, Washington (2001)
17. Jeong, W.-K., Kim, C.-H.: Direct reconstruction of a displaced subdivision surface from unorganized points. Graph. Models **64**(2), 78–93 (2002)
18. Marinov, M., Kobbelt, L.: Optimization methods for scattered data approximation with subdivision surfaces. Graph. Models **67**(5), 452–473 (2005)
19. Halstead, M., Kass, M., DeRose, T.: Efficient, fair interpolation using Catmull–Clark surfaces. In: SIGGRAPH '93, pp. 35–44 (1993)
20. Grimm, C., Laidlaw, D., Crisco, J.: Fitting manifold surfaces to 3d point clouds. J. Biomech. Eng. **124**, 136–140 (2002)
21. Ying, L., Zorin, D.: A simple manifold-based construction of surfaces of arbitrary smoothness. ACM Trans. Graph. **23**, 271–275 (2004)
22. Navau, J.C., Garcia, N.P.: Modeling surfaces from meshes of arbitrary topology. Comput. Aided Geom. Des. **17**, 643–671 (2000) ISSN: 0167-8396
23. Grimm, C.: Parameterization using manifolds. Int. J. Shape Model. **10**, 51–80 (2004)
24. Saba, S., Yavneh, I., Gotsman, C., Sheffer, A.: Practical spherical embedding of manifold triangle meshes. In: Shape Modelling International, pp. 256–265, June 2005
25. Erickson, J., Whittlesey, K.: Greedy optimal homotopy and homology generators. In: SODA, pp. 1038–1046 (2005)
26. Grimm, C., Hughes, J.: Parameterizing n-holed tori. In: Mathematics of Surfaces X, pp. 14–29, September 17–19, 2003
27. Sander, P.V., Gortler, S.J., Snyder, J., Hoppe, H.: Signal-specialized parametrization. In: EGRW '02, pp. 87–98. Eurographics Association, 2002
28. LLoyd, S.: Least square quantization in pcm. IEEE Trans. Inf. Theory 129–137 (1982)
29. Marai, G.E., Grimm, C., Laidlaw, D.: Arthrodial joint markerless cross-parameterization and biomechanical visualization. IEEE Trans. Vis. Comput. Graph. (2007)

Cindy Grimm is an associate professor at Washington University in St. Louis, working in the area of Computer Graphics. Her specific research interests are in art-based rendering and surface modeling for biomedical applications. She received her PhD from Brown University in 1996.

Tao Ju is an Assistant Professor in the Department of Computer Science and Engineering at the Washington University in St. Louis (USA). He obtained his MSc and PhD degrees in Computer Science at Rice University in 2005. He conducts research in computer graphics and biomedical applications, and is particularly interested in geometric modeling and processing.

Ly Phan received the BA in English from Ho Chi Minh City University of Social Sciences and Humanities, Vietnam, and the BSc in Computer Science from Southern Arkansas University, Arkansas. She is currently pursuing the PhD degree in Computer Science at Washington University in St. Louis. She was awarded a research internship under the Distributed Mentor Program (CRA-W) in 2005 and the Imaging Sciences Pathway Fellowship 2009–2010. Her research interests are in computer graphics and medical imaging. Her work includes matching, parameterization, and reconstruction of surfaces.

John Hughes (BA, Mathematics, Princeton, 1977; PhD, Mathematics, U.C. Berkeley, 1982) is a Professor of Computer Science at Brown University. His research is in computer graphics, particularly those aspects of graphics involving substantial mathematics. As author or co-author of 19 SIGGRAPH papers, he has done research in geometric modeling, user interfaces for modeling, non-photorealistic rendering, and animation systems. He has served as an associate editor for ACM Transaction on Graphics and the Journal of Graphics Tools, and has been on the SIGGRAPH program committee multiple times. He was the Papers Chair for SIGGRAPH 2002, and is Chair of SIGGRAPH's Technical Awards Committee. He is a co-author of "Computer Graphics: Principles and Practice," a standard reference work.