Submitted for Publication, January 2000

Normal Meshes

Igor Guskov* Caltech
Kiril Vidimče† Mississippi State University
Wim Sweldens‡ Bell Laboratories
Peter Schröder§ Caltech

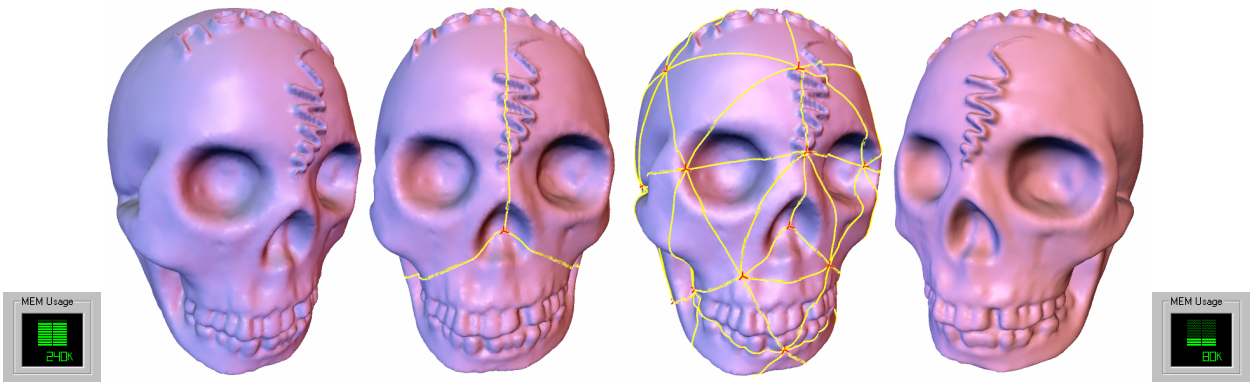

Figure 1: *Left: original mesh (3 floats/vertex). Middle: two stages of our algorithm. Right: normal mesh (1 float/vertex). (Skull dataset courtesy Headus, Inc.)*

Abstract

Normal meshes are new fundamental surface descriptions inspired by differential geometry. A normal mesh is a multiresolution mesh where each level can be written as a normal offset from a coarser version. Hence the mesh can be stored with a single float per vertex. We present an algorithm to approximate any surface arbitrarily closely with a normal semi-regular mesh. Normal meshes are useful in numerous applications such as compression, filtering, rendering, texturing, and modeling.

1 Introduction

The standard way to parameterize a surface involves *three* scalar functions $x(u,v)$, $y(u,v)$, $z(u,v)$. Yet differential geometry teaches us that smooth surfaces locally can be described by a *single* scalar height function over the tangent plane. Loosely speaking one can say that the geometric information of a surface can be contained in only a single dimension, the height over this plane. This observations holds infinitesimally; only special cases such as terrains and star-shaped surfaces can globally be described with a single function.

In practice we often approximate surfaces using a triangle mesh. While describing meshes is relatively easy, they have lost much of the structure inherent in the original surface. For example, the above observation that locally a surface can be characterized by a scalar function is not reflected in the fact that we store 3 floats per vertex. In other words, the correlation between neighboring sample locations implied by the smoothness assumption is not reflected, leading to an inherently redundant representation.

While vertex locations come as 3-dimensional quantities, the above considerations tell us that locally two of those dimensions represent parametric information and only the third captures geometric, or shape, information. For a given smooth shape one may choose different parameterizations, yet the geometry remains the same. In the case of a mesh we can observe this by noticing that infinitesimal tangential motion of a vertex does not change the geometry, only the sampling pattern, or parameterization. Moving in the normal direction on the other hand changes the geometry and leaves parameter information undisturbed.

1.1 Goals and Contributions

Based on the above observations, the aim of the present paper is to compute mesh representations that only require a single scalar per vertex. We call such representations *normal meshes*. The main insight is that this can be done using multiresolution and local frames. A normal mesh has a hierarchical representation so that all detail coefficients when expressed in local frames are scalar, i.e., they only have a normal component. In the context of compression, for example, this implies that parameter information can be perfectly predicted and residual error is entirely constrained to the normal direction, i.e., contains only geometric information. Note that because of the local frames normal mesh representations are non-linear.

Of course we cannot expect a given arbitrary input mesh to possess a hierarchical representation which is normal. Instead we describe an algorithm which takes an arbitrary topology input mesh

*ivguskov@cs.caltech.edu
†vkire@cs.msstate.edu
‡wim@lucent.com
§ps@cs.caltech.edu

and produces a semi-regular normal mesh describing the same geometry. Aside from a small amount of base domain information, *our normal mesh transform converts an arbitrary mesh from a 3 parameter representation into a purely scalar representation*. We demonstrate our algorithm by applying it to a number of models and experimentally characterize some of the properties which make normal meshes so attractive for computations.

The study of normal meshes is of interest for a number of reasons: they

- bring our computational representations back towards the "first principles" of differential geometry;
- are very storage and bandwidth efficient, describing a surface as a succinctly specified base shape plus a hierarchical normal map;
- are an excellent representation for compression since all variance is "squeezed" into a single dimension.

1.2 Related Work

Efficient representations for irregular connectivity meshes have been pursued by a number of researchers. This research is motivated by our ability to acquire densely sampled, highly detailed scans of real world objects [16] and the need to manipulate these efficiently. Semi-regular—or subdivision connectivity—meshes offer many advantages over the irregular setting due of their well developed mathematical foundations and data structure simplicity [20]; many powerful algorithms require their input to be in semi-regular form [18, 19, 22, 1]. This has led to the development of a number of algorithms to convert existing irregular meshes to semi-regular form through remeshing. Eck et al. [8] use Voronoi tiling and harmonic maps to build a parameterization and remesh onto a semi-regular mesh. Krischnamurthy and Levoy [13] demonstrated user driven remeshing for the case of bi-cubic patches, while Lee et al. [15] proposed an algorithm based on feature driven mesh reduction to develop smooth parameterizations of meshes in an automatic fashion. These methods use the parameterization subsequently for semi-regular remeshing.

Our work is related to these approaches in that we also construct a semi-regular mesh from an arbitrary connectivity input mesh. However, in previous work prediction residuals, or detail vectors, were not optimized to have properties such as normality. The main focus was on the establishment of a smooth parameterization which was then semi-regularly sampled.

The discussion of parameter versus geometry information originates in the work done on irregular curve and surface subdivision [4] [12] and intrinsic curvature normal flow [5]. There it is shown that unless one has the correct parameter side information, it is not possible to build an irregular smooth subdivision scheme. While such schemes are useful for editing and texturing applications, they cannot be used for succinct representations because the parameter side-information needed is excessive. In the case of normal meshes these issues are entirely circumvented in that all parameter information vanishes and the mesh is reduced to purely geometric, i.e., scalar in the normal direction, information.

Finally, we mention the connection to displacement maps [3], and in particular normal displacement maps. These are popular for modeling purposes and used extensively in high end rendering systems such as RenderMan. In a sense we are solving here the associated inverse problem. Given some geometry, find a simpler geometry and a set of normal displacements which together are equivalent to the original geometry. Typically, normal displacement maps are single level, whereas we aim to build them in a fully hierarchical way. For example, single level displacements maps were used in [13] to capture the fine detail of a 3D photography model. Cohen et al. [2] sampled normal fields of geometry and maintained these in texture maps during simplification. While these approaches all differ significantly from our interests here, it is clear that maps of this and related nature are of great interest in many contexts.

2 Normal Polylines

Before we look at surfaces and normal meshes, we introduce some of the concepts using curves and normal polylines. A curve in the plane is described by a pair of parametric functions $\mathbf{s}(t) = (x(t), y(t))$ with $t \in [0, 1]$. We would like to describe the points on the curve with a single scalar function. In practice one uses polylines to approximate the function. Let $\mathbf{l}(\mathbf{p}, \mathbf{p}')$ be the linear segment between the points $\mathbf{p}$ and $\mathbf{p}'$. A standard way to build a polyline multiresolution approximation is to sample the curve at points $\mathbf{s}_{j,k}$ where $\mathbf{s}_{j,k} = \mathbf{s}_{j+1,2k}$ and define the jth level approximation as

$$\mathbf{L}_j = \bigcup_{0 \le k < 2^j} \mathbf{l}(\mathbf{s}_{j,k}, \mathbf{s}_{j,k+1}).$$

To move from $\mathbf{L}_j$ to $\mathbf{L}_{j+1}$ we need to insert the points $\mathbf{s}_{j+1,2k+1}$ (Figure 2, left). Clearly this requires two scalars: the two coordinates of $\mathbf{s}_{j+1,2k+1}$. Alternatively one could compute the difference $\mathbf{s}_{j+1,2k+1} - \mathbf{m}$ between the new point and some predicted point $\mathbf{m}$, say the midpoint of the neighboring points $\mathbf{s}_{j,k}$ and $\mathbf{s}_{j,k+1}$. This detail has a tangential component $\mathbf{m} - \mathbf{b}$ and a normal component $\mathbf{b} - \mathbf{s}_{j+1,2k+1}$. The normal component is the *geometric* information while the tangential component is the *parameter* information. The way to build polylines that can be described with one

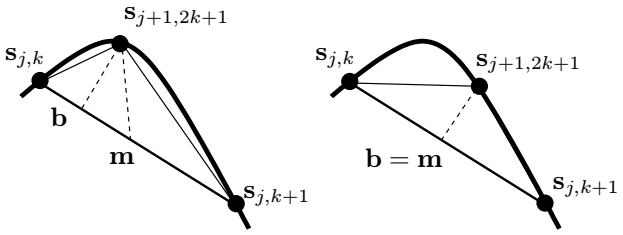

Figure 2: *Removing one point $\mathbf{s}_{j+1,2k+1}$ in a polyline multiresolution and recording the difference with the midpoint $\mathbf{m}$. On the left a general polyline where the detail has both a normal and a tangential component. On the right a normal polyline where the detail is purely normal.*

scalar per point, is to make sure that the parameter information is always zero, i.e., $\mathbf{b} = \mathbf{m}$, see Figure 2, right. If the triangle $\mathbf{s}_{j,k}$, $\mathbf{s}_{j+1,2k+1}$, $\mathbf{s}_{j,k+1}$ is Isosceles, there is no parameter information. Consequently we say that a polyline is normal if a multiresolution structure exists where every removed point forms an Isosceles triangle with its neighbors. Then there is zero parameter information and the polyline can be represented with one scalar per point, namely the normal component of the associated detail.

For a general polyline the removed triangles are hardly ever exactly Isosceles and the polyline hence is not normal. Below we describe a procedure to build a normal polyline approximation for any continuous curve. The easiest is to start building Isosceles triangles from the coarsest level. Start with the first base $\mathbf{l}(\mathbf{s}_{0,0}, \mathbf{s}_{0,1})$, see Figure 3. Next take its midpoint and check where the normal direction crosses the curve. Because the curve is continuous, there has to be at least one such point. If there are multiple pick any one. Call this point $\mathbf{s}_{1,1}$ and define the first triangle. Now repeat this process. Each time $\mathbf{s}_{j+1,2k+1}$ is found where the normal to the midpoint of $\mathbf{s}_{j,k}$ and $\mathbf{s}_{j,k+1}$ crosses the curve. Thus any continuous curve can be approximated arbitrarily closely with a normal polyline. The result is a series of polylines $\mathbf{L}_j$ all of which are normal with respect to midpoint prediction. Effectively each level is

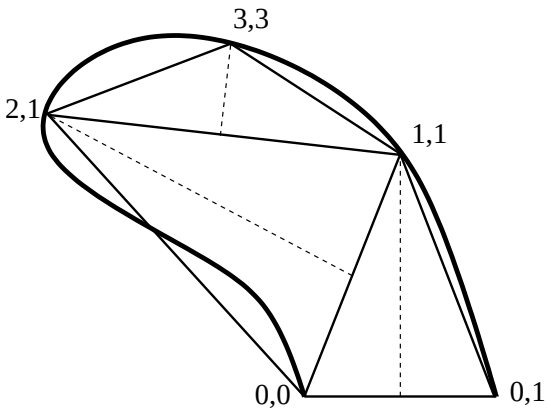

Figure 3: *Construction of a normal polyline. We start with the coarsest level and each time check where the normal to the midpoint crosses the curve. For simplicity only the indices of the $\mathbf{s}_{j,k}$ points are shown and only certain segments are subdivided. The polyline $(0,0)-(2,1)-(3,3)-(1,1)-(0,1)$ is determined by its endpoints and three scalars, the heights of the Isosceles triangles.*

parameterized with respect to the one coarser level. Because the polylines are normal, only a single scalar value, the normal component, needs to be recorded for each point. We have a polyline with no parameter information.

One can also consider normal polylines with respect to fancier predictors. For example one could compute a base point and normal estimate using the well known 4 point rule. Essentially any predictor which only depends on the coarser level is allowed. For example one can also use irregular schemes [4]. Also one does not need to follow the standard way of building levels by downsampling every other point, but instead could take any ordering. This leads to the following definition of a normal polyline:

Definition 1 *A polyline is normal if a removal order of the points exist such that each removed point lies in the normal direction from a base point, where the normal direction and base point only depend on the remaining points.*

Hence a normal polyline is completely determined by a scalar component per vertex.

Normal polylines are closely related to certain well known fractal curves such as the Koch Snowflake[1], see Figure 4. Here each time a line segment is divided into three subsegments. The left and right get a normal coefficient of zero, while the middle receives a normal coefficient such that the resulting triangle is equilateral. Hence the polylines leading to the snowflake are normal with respect to midpoint subdivision.

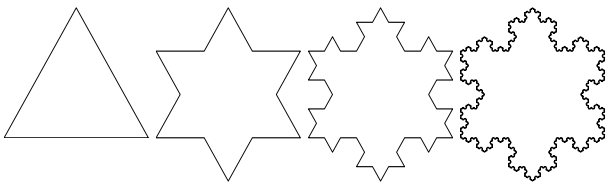

Figure 4: *Four normal polylines converging to the Koch snowflake.*

There is also a close connection with wavelets. The normal coefficients can be seen as a piecewise linear wavelet transform of the original curve. Because the tangential components are always zero there are half as many wavelet coefficients as there are original scalar coefficients. Thus one saves 50% memory right away. In addition of course the wavelets have their usual decorrelation properties. In the functional case the above transform corresponds to an unlifted interpolating piecewise linear wavelet transform as introduced by Donoho [6]. There it is shown that interpolating wavelets with no primal, but many dual moments are well suited for smooth functions. Unlike in the function setting, not all wavelets from the same level j have the same physical scale. Here the scale of each coefficient is essentially the length of the base of its Isosecles triangle. To obtain a scale ordered decomposition of a normal curve one would sort all the coefficients in order of decreasing base length.

[1] Niels Fabian Helge von Koch (Sweden, 1870-1924)

3 Normal Meshes

We begin by establishing terminology. A triangle mesh $\mathcal{M}$ is a pair $(\mathcal{P}, \mathcal{K})$, where $\mathcal{P}$ is a set of N point positions $\mathcal{P} = \{\mathbf{p}_i = (x_i, y_i, z_i) \in \mathbf{R}^3 \mid 1 \leq i \leq N\}$, and $\mathcal{K}$ is an *abstract simplicial complex* which contains all the topological, i.e., adjacency information. The complex $\mathcal{K}$ is a set of subsets of $\{1, \ldots, N\}$. These subsets come in three types: vertices $\{i\}$, edges $\{i, j\}$, and faces $\{i, j, k\}$. Two vertices i and j are *neighbors* if $\{i, j\} \in \mathcal{E}$. The 1-ring neighbors of a vertex i form a set $\mathcal{V}(i) = \{j \mid \{i, j\} \in \mathcal{E}\}$.

We can derive a definition of normal triangle meshes inspired by the curve case. Consider a hierarchy of triangle meshes $\mathcal{M}_j$ built using mesh simplification with vertex removals. These meshes are nested in the sense that $\mathcal{P}_j \subset \mathcal{P}_{j+1}$. Take a removed vertex $\mathbf{p}_i \in \mathcal{P}_{j+1} \setminus \mathcal{P}_j$. For the mesh to be normal we need to be able to find a base point $\mathbf{b}$ and normal direction N that only depend on $\mathcal{P}_j$, so that $\mathbf{p}_i - \mathbf{b}$ lies in the direction N. This leads to the following definition.

Definition 2 *A mesh $\mathcal{M}$ is normal in case a sequence of vertex removals exists so that each removed vertex lies on a line defined by a base point and normal direction which only depends on the remaining vertices.*

Thus a normal mesh can be described by a small base domain and one scalar coefficient per vertex.

As in the curve case, a mesh is in general not normal. The chance that the difference between a removed point and a predicted base point lies exactly in a direction that only depends on the remaining vertices is essentially zero. Hence the only way to obtain a normal mesh is to change the triangulation. We decide to use semi-regular meshes, i.e., meshes whose connectivity is formed by successive quadrisection of coarse base domain faces.

As in the curve setting, the way to build a normal mesh is to start from the coarse level or base domain. For each new vertex we compute a base point as well as a normal direction and check where the line defined by the base point and normal intersects the surface. The situation, however, is much more complex than in the curve case for two reasons: (1) There could be no intersection point. (2) There could be many intersection points, but only one correct one.

In case there are no intersection points, strictly speaking no fully normal mesh can be built from this base domain. If that happens, we relax the definition of normal meshes some and allow a small number of cases where the new points do not lie in the normal direction. Thus the algorithm needs to find a suitable non-normal location for the new point. In case there are many intersection points the algorithm needs to figure out which one is the right one. If the wrong one is chosen the normal mesh will start folding over itself or leave creases. Any algorithm which blindly picks an intersection point is doomed.

Parameterization In order to find the right piercing point or suggest a good alternate, one need to be able to easily navigate

around the surface. The way to do this is to build a smooth parameterization of the surface region of interest. This is a basic building block of our algorithm. Several parameterization methods have been proposed and our method takes components from each of them: mesh simplification and polar maps from MAPS [15], patchwise relaxation from [8], and a specific smoothness functional similar to the one used in [9] and [17]. The algorithm will use numerous local parameterizations which need to be computed fast and robustly. Most of them are temporary and are quickly discarded unless they can be used as a starting guess for another parameterization.

Consider a region $\mathcal{R}$ of the mesh homeomorphic to a disc that we want to parameterize onto a convex planar region $\mathcal{B}$, i.e., find a bijective map $u : \mathcal{R} \to \mathcal{B}$. The map u is fixed by a boundary condition $\partial\mathcal{R} \to \partial\mathcal{B}$ and minimizes a certain energy functional. Several functionals can be used leading to, e.g., conformal or harmonic mappings. We take an approach based on minimizing second order differences which can be seen as a higher order discretization of the Laplacian. In short, the function u needs to satisfy the following equation in the interior:

$$u(\mathbf{p}_i) = \sum_{k \in \mathcal{V}(i)} \alpha_{ik} u(\mathbf{p}_k), \tag{1}$$

where $\mathcal{V}(i)$ is the 1-ring neighborhood of the vertex i and the weights α_{ik} come from the parameterization scheme introduced by Floater [9], see also [12]. The main advantage of the Floater weights is that they are always positive, which, combined with the convexity of the parametric region, guarantees that no triangle flipping can occur within the parametric domain. This is crucial for our algorithm. Note that this is not true in general for harmonic maps which can have negative weights. We use the iterative biconjugate gradient method [11] to obtain the solution to the system (1). Given that we often have a good starting guess this converges quickly.

Algorithm Our algorithm consists of 7 stages which are described below, some of which are shown for the molecule model in Figure 5. The molecule is a highly detailed and curved model. Any naive procedure for finding normal meshes is very unlikely to succeed.

1. Mesh simplification: We use the Garland-Heckbert [10] simplification based on half-edge collapses to create a mesh hierarchy $(\mathcal{P}_j, \mathcal{K}_j)$. We use the coarsest level $(\mathcal{P}_0, \mathcal{K}_0)$ as an initial guess for our base domain $(\mathcal{Q}_0, \mathcal{K}_0)$. The first image of Figure 5 shows the base domain for the molecule.

2. Building an initial net of curves: The purpose of this step is to connect the vertices of the base domain with a net of non intersecting curves on the different levels of the mesh simplification hierarchy. This can easily be done using the MAPS parameterization [15]. MAPS uses polar maps to build a bijection between a 1-ring and its retriangulation after the center vertex is removed. The concatenation of these maps is a bijective mapping between different levels $(\mathcal{P}_j, \mathcal{K}_j)$ in the hierarchy. The desired curves are simply the image of the base domain edges under this mapping. Because of the bijection no intersection can occur. Note that the curves start and finish at a vertex of the base domain, but need not follow the edges of the finer triangulation, i.e., they can cut across triangles. These curves define a network of triangular shaped patches corresponding to the base domain triangles. Later we will adjust these curves on some intermediate level and again use MAPS to propagate these changes to other levels. The top middle image of Figure 5 shows these curves for some intermediate level of the hierarchy.

3. Fixing the global vertices: A normal mesh is almost completely determined by the base domain. One has to choose the base domain vertices $\mathcal{Q}_0$ very carefully to reduce the number of non-normal vertices to a minimum. The coarsest level of the mesh simplification $\mathcal{P}_0$ is only a first guess. In this section we describe a procedure for repositioning the global vertices $\mathbf{q}_i$ with $\{i\} \in \mathcal{K}_0$. We impose the constraint that the $\mathbf{q}_i$ needs to coincide with some vertex $\mathbf{p}_k$ of the original mesh, but not necessarily $\mathbf{p}_i$.

The repositioning is typically done on some intermediate level j. Take a base domain vertex $\mathbf{q}_i$. We build a parameterization from the patches incident to vertex $\mathbf{q}_i$ to a disk in the plane, see Figure 6. Boundary conditions are assigned using arclength parameterization and parameter coordinates are iteratively computed for each level j vertex inside the shaded region. It is now easy to replace the point $\mathbf{q}_i$ with any level point from $\mathcal{P}_j$ in the shaded region. In particular we let the new $\mathbf{q}_i'$ be the point of $\mathcal{P}_j$ that in the parameter domain is closest to the center of the disk.

Once a new position $\mathbf{q}_i'$ is chosen, the curves can be redrawn by taking the inverse mapping of straight lines from the new point in the parameter plane. One can keep iterating this procedure, but we found that if suffices to cycle once through all base domain vertices.

We also provide for a user controlled repositioning. Then the user can replace the center vertex with any $\mathcal{P}_j$ point in the shaded region. The algorithm again uses the parameterization to recompute the curves from that point.

The top right of Figure 5 shows the repositioned vertices. Notice how some of them like the rightmost one have moved considerably.

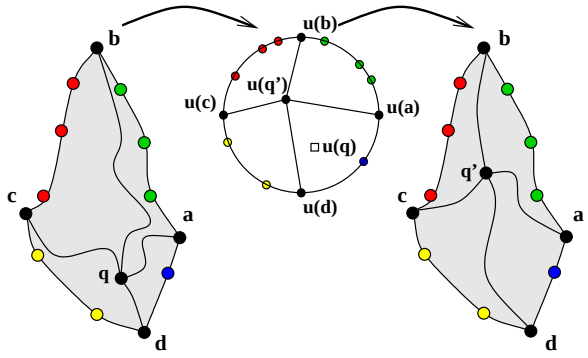

Figure 6: *Base domain vertex repositioning. Left: original patches around $\mathbf{q}_i$, middle: parameter domain, right: repositioned $\mathbf{q}_i$ and new patch boundaries. This is replaced with the vertex whose parameter coordinate are the closest to the center. The inverse mapping (right) is used to find the new position $\mathbf{q}_i'$ and the new curves.*

4. Fixing the global edges: The image of the global edges on the finest level will later be the patch boundaries of the normal mesh. For this reason we need to improve the smoothness of the associated curves at the finest level. We use a procedure similar to [8]. For each base domain edge $\{i, k\}$ we consider the region formed on the finest level mesh by its two incident patches. Let l and m be the opposing global vertices. We then compute a parameter function ρ within the diamond-shaped region of the surface. The boundary condition is set as $\rho(\mathbf{q}_i) = \rho(\mathbf{q}_k) = 0$, $\rho(\mathbf{q}_l) = 1$, $\rho(\mathbf{q}_m) = -1$, with linear variation along the edges. We then compute the parameterization and let its zero level set be our new curve. Again one could iterate this procedure till convergence but in practice one cycle suffices. The curves of the top right image in Figure 5 are the result of the curve smoothing on the finest level.

5. Initial parameterization: Once the global vertices and edges are fixed, one can start filling in the interior. This is done by computing the parameterization of each patch to a triangle while

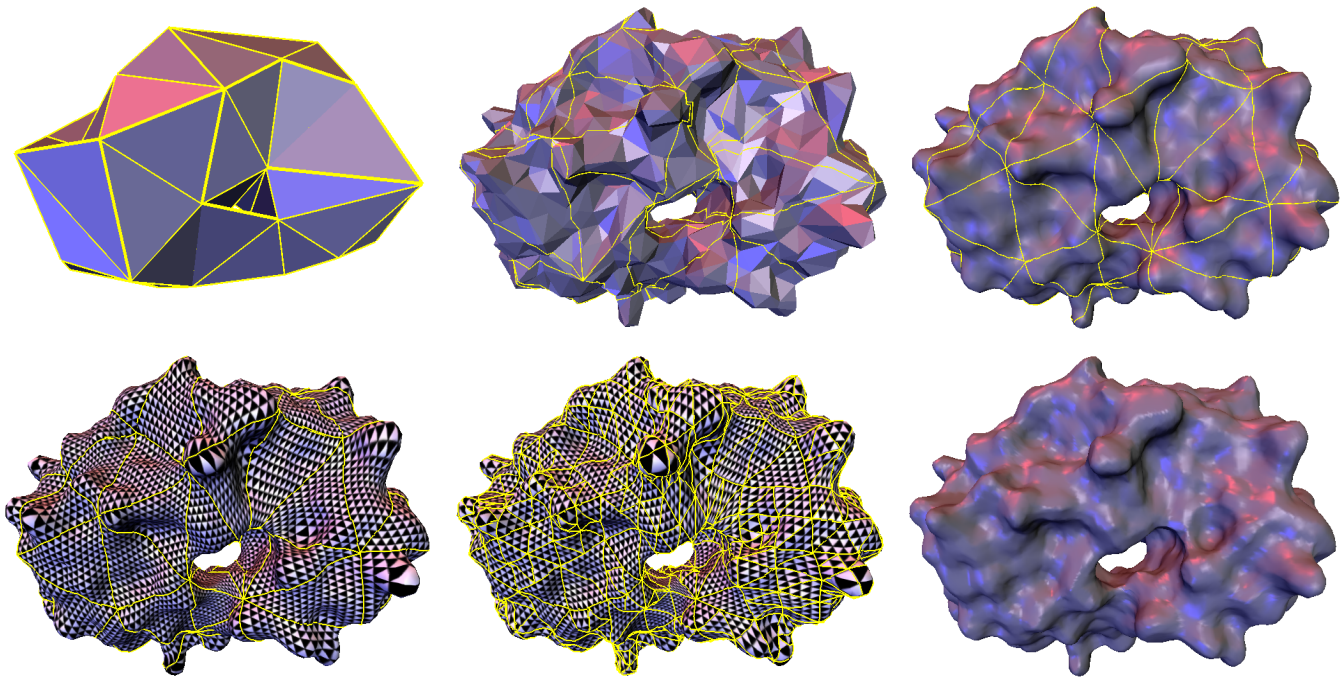

Figure 5: *The entire procedure shown for the molecule model. 1. Base domain. 2. Initial set of curves. 3. Global vertex repositioning 4. Initial Parameterization 5. Adjusting parameterization 6. Final normal mesh. (HIV protease surface model courtesy of Arthur Olson, The Scripps Research Institute)*

keeping the boundary fixed. The parameter coordinates from the last stage can serve as a good initial guess. We now have a smooth global parameterization. This parameterization is shown in the bottom left of Figure 5. Each triangle is given a triangular checkerboard texture to illustrate the parameterization.

6. Piercing: In this stage of the algorithm we start building the actual normal mesh. The canonical step is for a new vertex of the semi-regular mesh to find its position on the original mesh. In quadrisection every edge of level j generates a new vertex on level $j+1$. We first compute a base point using interpolating Butterfly subdivision [7] [21] as well as an approximation of the normal. This defines a straight line. This line may have multiple intersection points in which case we need to find the right one, or it could have none, in which case we need to come up with a good alternate.

Suppose that we need to produce the new vertex $\mathbf{q}$ that lies halfway along the edge $\{\mathbf{a}, \mathbf{c}\}$ with incident triangles $\{\mathbf{a}, \mathbf{c}, \mathbf{b}\}$ and $\{\mathbf{c}, \mathbf{a}, \mathbf{d}\}$, see Figure 7. Let the two incident patches form the region $\mathcal{R}$.

Build the straight line L defined by the base point $\mathbf{s}$ predicted by the Butterfly subdivision rule and the direction of the normal computed from the coarser level points. We find all the intersection points of L with the region $\mathcal{R}$ by checking all triangles inside.

If there is no intersection we take the point $\mathbf{v}$ that lies midway between the points $\mathbf{a}$ and $\mathbf{c}$ in the parameter domain: $u(\mathbf{v}) = (u(\mathbf{a}) + u(\mathbf{c}))/2$. This is the same point a standard parameterization based remesher would use.

In the case when there exist several intersections of the mesh region $\mathcal{R}$ with the piercing line L we choose the intersection point that is closest to the point $u(\mathbf{v})$ in the parameter domain. Let us denote by $u(\mathbf{q})$ the parametric coordinates of that piercing point. We accept this point as a valid point of the semi-regular mesh if $\|u(\mathbf{q}) - u(\mathbf{v})\| < \kappa \|u(\mathbf{a}) - u(\mathbf{v})\|$, where κ is an "aperture" parameter that specifies how much the parameter value of a pierced

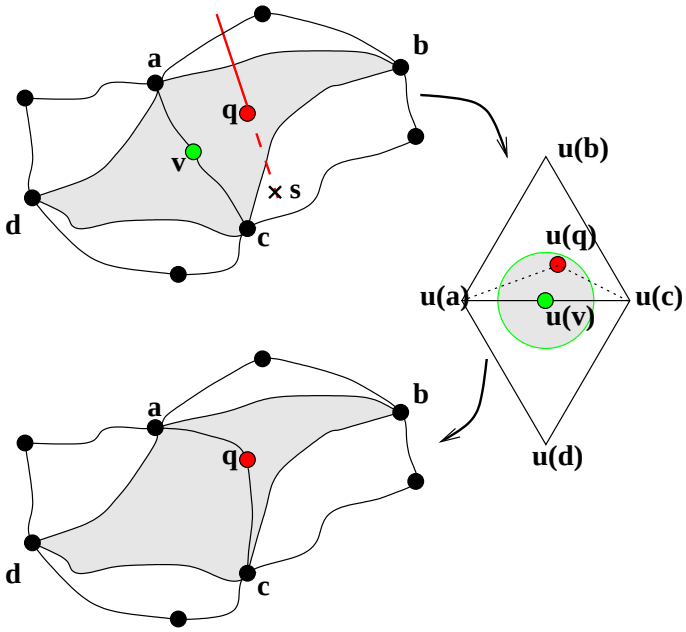

Figure 7: *Upper left: piercing, the Butterfly point is* $\mathbf{s}$*, the surface is pierced at the point* $\mathbf{q}$*, the parametrically suggested point* $\mathbf{v}$ *lies on the curve separating two regions of the mesh. Right: parameter domain, the pierced point falls inside the aperture and gets accepted. Lower left: the parameterization is adjusted to let the curve pass through* q*.*

point is allowed to deviate from the center of the diamond. Otherwise, the piercing point is rejected and the mesh takes the point with the parameter value $u(\mathbf{v})$.

7. Adjusting the parameterization: Once we have a new piercing point, we need to adjust the parameterization to reflect this. Essentially, the adjusted parameterization u should be such that the piercing point has the parameters $u(\mathbf{v}) =: u(\mathbf{q})$. When imposing such an isolated point constraint on the parameterization, there is no mathematical guarantee against flipping. Hence we draw a new piecewise linear curve through $u(\mathbf{q})$ in the parameter domain. This gives a new curve on the surface which passes through $\mathbf{q}$, see Figure 7. We then recompute the parameterization for each of the patches onto a triangle separately. We use a piecewise linear boundary condition with the half point at $\mathbf{q}$ on the common edge.

When all the new midpoints for the edges of a face of level j are computed, we can build the faces of level $j+1$. This is done by drawing three new curves inside the corresponding region of the original mesh, see Figure 8. Before that operation happens we need to ensure that a valid parameterization is available within the patch. The patch is parameterized onto a triangle with three piecewise linear boundary conditions each time putting the new points at the midpoint. Then the new points are connected in the parameter domain which allows us to draw new finer level curves on the original mesh. This produces a metamesh similar to [14] which replicates the structure of the semi-regular hierarchy on the surface of the original. The construction of the semi-regular mesh can be done adaptively with the error driven procedure from MAPS [15]. An example of parameterization adjustment after two levels of adaptive subdivision is shown in the bottom middle of Figure 5.

As the regions for which we compute parameterizations become smaller, the starting guesses are better and the solver becomes faster and faster. We use lazy parameter computation—the relaxation is run just before we actually need to use parameters for either a point location or a surface curve drawing procedure.

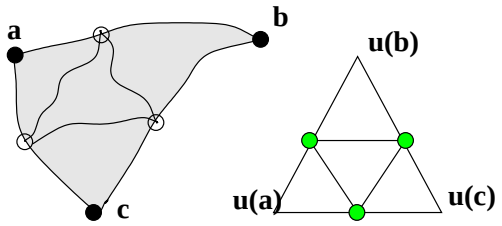

Figure 8: *Face split: Quadrisection in the parameter plane (left) leads to three new curves within the triangular patch (right).*

The aperture parameter κ of the piercing procedure provides control over how much of the original parameterization is preserved in the final mesh. At $\kappa = 0$ we build a mesh entirely based on the original global parameterization. At $\kappa = 1$ we attempt to build a purely normal mesh independent of the parameterization. In our experience, the best results were achieved when the aperture was set low (0.2) at the coarsest levels, and then increased to (0.6) on finer levels. On the very fine levels of the hierarchy, where the geometry of the semi-regular meshes closely follows the original geometry, one can often simply use a naive piercing procedure without parameter adjustment.

One may wonder if the continuous readjustment of parameterizations is really necessary. We have tried the naive piercing procedure without parameterization from the base domain and found that it typically fails on all models. An example is Figure 9 which shows 4 levels of naive piercing for the torus starting from a 102 vertex base mesh. Clearly, there are several regions with flipped and self-intersecting triangles. The error is about 20 times larger than the true normal mesh.

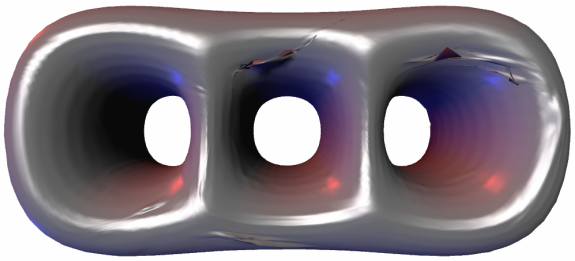

Figure 9: Naive piercing procedure. Clearly, several regions have flipped triangles and are self-intersecting.

Dataset	Size	Base	Normal mesh size	Not normal (%)	% L^2 error	Time (min)
Feline	49864	156	40346	729 (1.8%)	.015	4
Feline	49864	156	161790	729 (0.5%)	.004	6
Molecule	10028	37	9521	270 (2.8%)	.075	1.5
Molecule	10028	37	151515	270 (.02%)	.01	2
Rabbit	16760	33	8235	196 (2.4%)	.037	2
Rabbit	16760	33	32741	196 (0.6%)	.0067	2.5
Torus3	5884	98	5294	421 (8.0%)	.03	3
Torus3	5884	98	26002	436 (1.7%)	.01	4
Skull	20002	112	25376	817 (3.2%)	.02	2.5
Skull	20002	112	111376	817 (0.7%)	.007	3
Horse	48485	234	59319	644 (1.1%)	.004	6.8

Table 1: *Summary of normal meshing results for different models. For each model there were two runs performed—the first one shows the result of an adaptive normal mesh with the target number of vertices roughly equal to the original vertex count, the second one is a uniform normal mesh with mean-square error below 0.01%. The errors were computed with the I.E.I.-CNR Metro tool. The times are reported on a 700MHz Pentium III machine.*

4 Results

We have implemented the algorithms described in the preceding section, and performed a series of experiments in which normal meshes for various models were built. The summary of the results is given in Table 1. As we can see from the table, the normal semi-regular meshes have very high accuracy and hardly any non normal details.

One interesting feature of our normal meshing procedure is as follows: while the structure of patches comes from performing simplification there are far fewer restrictions on how coarse the base mesh can be. Note for example that the skull in Figure 1 was meshed with the tetrahedron as base mesh. This is largely due to the robust mesh parameterization techniques used in our approach.

Figure 10 shows normal meshes for rabbit, torus, and feline, as well as close-up of the skull (bottom left) and feline (bottom right) normal meshes. Note how smooth the meshes are across global edges and global vertices. This smoothness mostly comes from the normality, not the parameterization. It is thus an intrinsic quantity.

One of the most interesting observations coming from this work is that locally the normal meshes do not differ much from the non-normal ones, while offering huge benefits in terms of efficiency of representation. For example, Table 2 shows how the "aperture parameter" κ that governs the construction of normal meshes affects the number of detail coefficients with non-trivial tangential components for the model of the three hole torus (these numbers are typical for other models as well). In particular, we see that already a very modest acceptance strategy ($\kappa = 0.2$) gets rid of more than 90% of the tangential components in the remeshed model, and the more aggressive strategies offer even more benefits without affecting the error of the representation.

κ	normal	error (10^{-4})
0	0%	1.02
0.2	91.9%	1.05
0.4	92.4%	1.04
best	98.3%	1.02

Table 2: *The relation between the acceptance strategy during the piercing procedure and the percentage of perfectly normal details in the hierarchy. The original model has 5884 vertices, all the normal meshes have 26002 vertices (4 levels uniformly), and the base mesh contained 98 vertices. The best strategy in the last line used $\kappa = 0.2$ on the first three levels and afterward always accepted the piercing candidates*

5 Summary and Conclusion

In this paper we introduce the notion of *normal meshes*. Normal meshes are multiresolution meshes in which vertices can be found in the normal direction, starting from some coarse level. Hence only one scalar per vertex needs to be stored. We presented a robust algorithm for computing normal semi-regular meshes of any input mesh and showed that it produces very smooth triangulations on a variety of input models.

It is clear that normal meshes have numerous applications. We briefly discuss a few.

Compression Usually a wavelet transform of a standard mesh has three components which need to be quantized and encoded. Information theory tells us that the more non uniform the distribution of the coefficients the lower the first order entropy. Having 2/3 of the coefficients exactly zero will further reduce the bit budget. From an implementation viewpoint, we can almost directly hook the normal mesh coefficients up to the best known scalar wavelet image compression code.

Filtering It has been shown that applications such as smoothing, enhancement, and denoising can simply be effected through a suitable scaling of wavelet coefficients. In a normal mesh any such algorithm will run three times as fast. Also large scaling coefficients in a standard mesh will introduce large tangential components leading to flipped triangles. In a normal mesh this is much less likely to happen.

Texturing Normal semi-regular meshes are very smooth inside patches, across global edges, and around global vertices even when the base domain is exceedingly coarse, cf. the skull model. The implied parameterizations are highly suitable for all types of mapping applications.

Rendering Normal maps are a very powerful tool for decoration and enhancement of otherwise smooth geometry. In particular in the context of bandwidth bottlenecks it is attractive to be able to download a normal map into hardware and only send smooth coefficient updates for the underlying geometry. The normal mesh transform effectively solves the associated inverse problem: construct a normal map for a given geometry.

The concept of normal meshes opens up many new areas of research.

- Our algorithm uses interpolating subdivision to find the base point. Building normal meshes with respect to approximating subdivision is not straightforward.
- The theoretical underpinnings of normal meshes need to be studied. Do continuous variable normal descriptions of surfaces exist? What about stability? What about connections with curvature normal flow which acts to reduce normal information?
- We only addressed semi-regular normal meshes here, while the definition allows for the more flexible setting of progressive irregular mesh hierarchies.
- Purely scalar compression schemes for geometry need to be compared with existing coders.
- Generalize normal meshes to higher dimensions where the potential storage reduction is even greater.

Acknowledgments This work was supported in part by NSF (ACI-9624957, ACI-9721349, DMS-9874082), Alias|wavefront, a Packard Fellowship, and a Caltech Summer Undergraduate Research Fellowship (SURF). Special thanks to Nathan Litke for his subdivision library, to Andrei Khodakovsky, Mathieu Desbrun, Adi Levin, Arthur Olson, and Zoë Wood for helpful discussions, Chris Johnson for the use of the Utah-SGI visual supercomputing resources, and to Cici Koenig for production help. Datasets are courtesy Cyberware, Headus, The Scripps Research Institute, and University of Washington.

References

[1] CERTAIN, A., POPOVIC, J., DEROSE, T., DUCHAMP, T., SALESIN, D., AND STUETZLE, W. Interactive Multiresolution Surface Viewing. *Proceedings of SIGGRAPH 96* (1996), 91–98.

[2] COHEN, J., OLANO, M., AND MANOCHA, D. Appearance-Preserving Simplification. *Proceedings of SIGGRAPH 98* (1998), 115–122.

[3] COOK, R. L. Shade trees. *Computer Graphics (Proceedings of SIGGRAPH 84) 18*, 3 (1984), 223–231.

[4] DAUBECHIES, I., GUSKOV, I., AND SWELDENS, W. Regularity of Irregular Subdivision. *Constr. Approx. 15* (1999), 381–426.

[5] DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. *Proceedings of SIGGRAPH 99* (1999), 317–324.

[6] DONOHO, D. L. Interpolating wavelet transforms. Preprint, Department of Statistics, Stanford University, 1992.

[7] DYN, N., LEVIN, D., AND GREGORY, J. A. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *ACM Transactions on Graphics 9*, 2 (1990), 160–169.

[8] ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution Analysis of Arbitrary Meshes. *Proceedings of SIGGRAPH 95* (1995), 173–182.

[9] FLOATER, M. S. Parameterization and Smooth Approximation of Surface Triangulations. *Computer Aided Geometric Design 14* (1997), 231–250.

[10] GARLAND, M., AND HECKBERT, P. S. Surface Simplification Using Quadric Error Metrics. In *Proceedings of SIGGRAPH 96*, 209–216, 1996.

[11] GOLUB, G. H., AND LOAN, C. F. V. *Matrix Computations*, 2nd ed. The John Hopkins University Press, Baltimore, 1983.

[12] GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. Multiresolution Signal Processing for Meshes. *Proceedings of SIGGRAPH 99* (1999), 325–334.

[13] KRISHNAMURTHY, V., AND LEVOY, M. Fitting Smooth Surfaces to Dense Polygon Meshes. *Proceedings of SIGGRAPH 96* (1996), 313–324.

[14] LEE, A. W. F., DOBKIN, D., SWELDENS, W., AND SCHRÖDER, P. Multiresolution Mesh Morphing. *Proceedings of SIGGRAPH 99* (1999), 343–350.

[15] LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. MAPS: Multiresolution Adaptive Parameterization of Surfaces. *Proceedings of SIGGRAPH 98* (1998), 95–104.

[16] LEVOY, M. The Digital Michelangelo Project. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling*, October 1999.

[17] LÉVY, B., AND MALLET, J. Non-Distorted Texture Mapping for Sheared Triangulated Meshes. *Proceedings of SIGGRAPH 98* (1998), 343–352.

[18] LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics 16*, 1 (1997), 34–73. Originally available as TR-93-10-05, October, 1993, Department of Computer Science and Engineering, University of Washington.

[19] SCHRÖDER, P., AND SWELDENS, W. Spherical Wavelets: Efficiently Representing Functions on the Sphere. *Proceedings of SIGGRAPH 95* (1995), 161–172.

[20] ZORIN, D., AND SCHRÖDER, P., Eds. *Subdivision for Modeling and Animation*. Course Notes. ACM SIGGRAPH, 1999.

[21] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interpolating Subdivision for Meshes with Arbitrary Topology. *Proceedings of SIGGRAPH 96* (1996), 189–192.

[22] ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive Multiresolution Mesh Editing. *Proceedings of SIGGRAPH 97* (1997), 259–268.

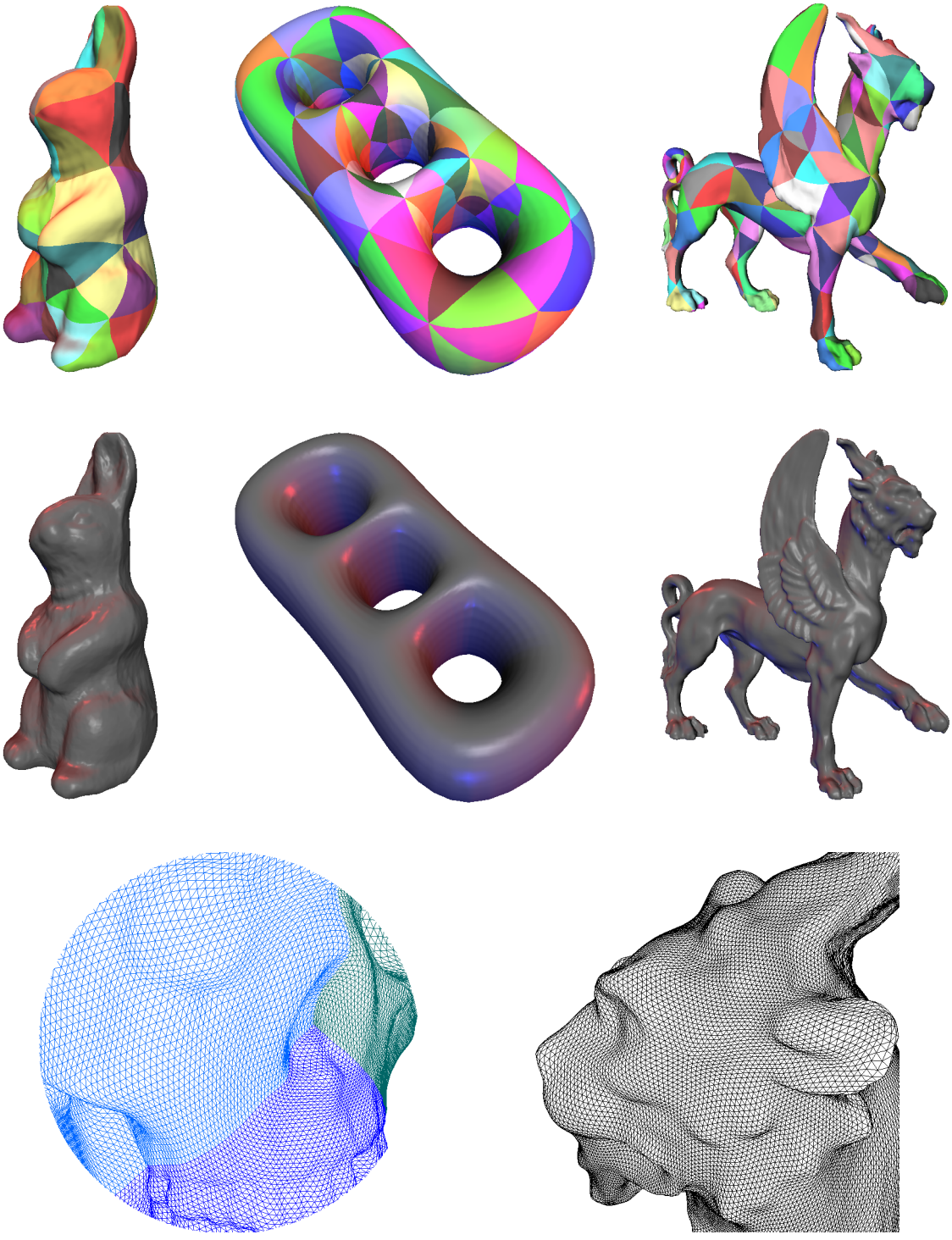

Figure 10: *Colorplate.*