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 can be useful in numerous applications such as compression, filtering, rendering, texturing, and modeling.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - curve, surface, solid, and object representations; hierarchy and geometric transformations; G.1.2 [Numerical Analysis]: Approximation - approximation of surfaces and contours, wavelets and fractals

Additional Keywords: Meshes, subdivision, irregular connectivity, surface parameterization, multiresolution, wavelets.

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 observation 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 de-
scribe an algorithm which takes an arbitrary topology input mesh 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 [19] 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 [23]; many powerful algorithms require their input to be in semi-regular form [21, 22, 25, 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. [9] use Voronoi tiling and harmonic maps to build a parameterization and remesh onto a semi-regular mesh. Krishnamurthy and Levoy [15] demonstrated user driven remeshing for the case of bi-cubic patches, while Lee et al. [18] 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] [13] 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 [15] 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.

In independent work, Lee et al. pursue a goal similar to ours [17]. They introduce displaced subdivision surfaces which can be seen as a two level normal mesh. Because only two levels are used, the base domain typically contains more triangles than in our case. Also the normal offsets are oversampled while in our case, the normal offsets are critically sampled.

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 \( s(t) = (x(t), y(t)) \) 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 \( I(p, p') \) be the linear segment between the points \( p \) and \( p' \). A standard way to build a polyline multiresolution approximation is to sample the curve at points \( s_{j,k} \) where \( s_{j,k} = s_{j+1,2k} \) and define the \( j \)th level approximation as

\[
L_j = \bigcup_{0 \leq k < 2^j} I(s_{j,k}, s_{j,k+1}).
\]

To move from \( L_j \) to \( L_{j+1} \) we need to insert the points \( s_{j+1,2k+1} \) (Figure 2, left). Clearly this requires two scalars: the two coordinates of \( s_{j+1,2k+1} \). Alternatively one could compute the difference \( s_{j+1,2k+1} - m \) between the new point and some predicted point \( m \), say the midpoint of the neighboring points \( s_{j,k} \) and \( s_{j,k+1} \). This detail has a tangential component \( m - b \) and a normal component \( b - s_{j+1,2k+1} \). 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 scalar per point, is to make sure that the parameter information is “squeezed” into a single dimension.

![Figure 2: Removing one point \( s_{j+1,2k+1} \) in a polyline multiresolution and recording the difference with the midpoint \( m \). 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.](image)

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.

In independent work, Lee et al. pursue a goal similar to ours [17]. They introduce displaced subdivision surfaces which can be seen as a two level normal mesh. Because only two levels are used, the base domain typically contains more triangles than in our case. Also the normal offsets are oversampled while in our case, the normal offsets are critically sampled.

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 \( s(t) = (x(t), y(t)) \) 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 \( I(p, p') \) be the linear segment between the points \( p \) and \( p' \). A standard way to build a polyline multiresolution approximation is to sample the curve at points \( s_{j,k} \) where \( s_{j,k} = s_{j+1,2k} \) and define the \( j \)th level approximation as

\[
L_j = \bigcup_{0 \leq k < 2^j} I(s_{j,k}, s_{j,k+1}).
\]

To move from \( L_j \) to \( L_{j+1} \) we need to insert the points \( s_{j+1,2k+1} \) (Figure 2, left). Clearly this requires two scalars: the two coordinates of \( s_{j+1,2k+1} \). Alternatively one could compute the difference \( s_{j+1,2k+1} - m \) between the new point and some predicted point \( m \), say the midpoint of the neighboring points \( s_{j,k} \) and \( s_{j,k+1} \). This detail has a tangential component \( m - b \) and a normal component \( b - s_{j+1,2k+1} \). 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 scalar per point, is to make sure that the parameter information is always zero, i.e., \( b = m \), see Figure 2, right. If the triangle \( s_{j,k}, s_{j+1,2k+1}, s_{j+1,2k+1} \) is Isosceles, there is no parameter information.

Consequently we say that a polyline is normal if a multiresolution approximation is to sample the curve at points \( s_{j,k} \) where \( s_{j,k} = s_{j+1,2k} \) and define the \( j \)th level approximation as

\[
L_j = \bigcup_{0 \leq k < 2^j} I(s_{j,k}, s_{j,k+1}).
\]

To move from \( L_j \) to \( L_{j+1} \) we need to insert the points \( s_{j+1,2k+1} \) (Figure 2, left). Clearly this requires two scalars: the two coordinates of \( s_{j+1,2k+1} \). Alternatively one could compute the difference \( s_{j+1,2k+1} - m \) between the new point and some predicted point \( m \), say the midpoint of the neighboring points \( s_{j,k} \) and \( s_{j,k+1} \). This detail has a tangential component \( m - b \) and a normal component \( b - s_{j+1,2k+1} \). 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 scalar per point, is to make sure that the parameter information is always zero, i.e., \( b = m \), see Figure 2, right. If the triangle \( s_{j,k}, s_{j+1,2k+1}, s_{j+1,2k+1} \) 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 hence the polyline 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 \( I(s_{0,0}, s_{0,1}) \), 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.
crosses the curve. For simplicity only the indices of the coarsest level and each time check where the normal to the midpoint of the curve between $s_{j,k}$ and $s_{j,k+1}$ crosses the portion of the curve between $s_{j,k}$ and $s_{j,k+1}$. Thus any continuous curve can be approximated arbitrarily closely with a normal polyline. The result is a series of polylines $L_j$ all of which are normal with respect to midpoint prediction. Effectively each level is 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 direction and check where the normal to the midpoint of the curve between $s_{j,k}$ and $s_{j,k+1}$ crosses the portion of the curve between $s_{j,k}$ and $s_{j,k+1}$. Thus any continuous curve can be approximated arbitrarily closely with a normal polyline. The result is a series of polylines $L_j$ all of which are normal with respect to midpoint prediction. Effectively each level is 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.

Call this point $s_{1,1}$ and define the first triangle. Now split the curve into two parts and repeat the procedure on each subcurve. Each time $s_{j+1,k+1}$ is found where the normal to the midpoint of $s_{j,k}$ and $s_{j,k+1}$ crosses the portion of the curve between $s_{j,k}$ and $s_{j,k+1}$.

The situation, however, is much more complex than in the curve case. Consider a hierarchy of triangle meshes $M_j$ built using mesh simplification with vertex removals. These meshes are nested in the sense that $P_j \subset P_{j+1}$. Take a removed vertex $p_i \in P_{j+1} \setminus P_j$. For the mesh to be normal we need to be able to find a base point $b$ and normal direction $N$ that only depend on $P_j$, so that $p_i - b$ lies in the direction $N$. This leads to the following definition.

**Definition 2** A mesh $M$ 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 needs 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 [18], patchwise relaxation from [9], and a specific smoothness functional similar to the one used in [10] and [20]. The algorithm will use 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 $u : \mathcal{R} \rightarrow \mathcal{B}$. The map $u$ is fixed by a boundary condition $\partial \mathcal{R} \rightarrow \partial \mathcal{B}$ 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 the work of Floater [10].

In short, the function $u$ needs to satisfy the following equation in the interior:

$$u(p_i) = \sum_{k \in V(i)} \alpha_{ik} u(p_k),$$

where $V(i)$ is the 1-ring neighborhood of the vertex $i$ and the weights $\alpha_{ik}$ come from the shape-preserving parameterization scheme [10]. 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 [12] 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 [11] simplification based on half-edge collapses to create a mesh hierarchy $(P_j, K_j)$. We use the coarsest level $(P_0, K_0)$ 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 [18]. 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 $(P_j, K_j)$ 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 $P_0$ is only a first guess. In this section we describe a procedure for repositioning the global vertices $\mathcal{Q}_0$ with $\{i\} \in \mathcal{K}_0$. We impose the constraint that the $q_i$ needs to coincide with some vertex $p_i$ of the original mesh, but not necessarily $p_i$.

The repositioning is typically done on some intermediate level $j$. Take a base domain vertex $q_i$. We build a parameterization from the patches incident to vertex $q_i$ 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 $q_i$ with any level point from $P_j$ in the shaded region. In particular we let the new $q_j'$ be the point of $P_j$ that in the parameter domain is closest to the center of the disk. The exact center of the disk, in general, does not correspond to a vertex of the mesh.

Once a new position $q_j'$ 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 $P_j$ 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.](image_url)

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 [9]. 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 $\rho$ within the diamond-shaped region of the surface. The boundary condition is set as $\rho(q_k) = \rho(q_l) = 0$, $\rho(q_j) = 1$, $\rho(q_0) = -1$, 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.

Notice that a similar result can be achieved by allowing the user to position the global vertices and draw the boundaries of the patches manually. Indeed, the following steps of the algorithm do not depend on how the initial net of surface curves is produced.
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 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 quadresection every edge of level $j$ generates a new vertex on level $j + 1$. We first compute a base point using interpolating Butterfly subdivision [8] [24] 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 $q$ that lies halfway along the edge $[a, c]$ with incident triangles $[a, c, b]$ and $[c, a, d]$, see Figure 7. Let the two incident patches form the region $\mathcal{R}$.

Build the straight line $L$ defined by the base point $s$ predicted by the Butterfly subdivision rule [24] 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 $v$ that lies midway between the points $a$ and $c$ in the parameter domain: $u(v) = (u(a) + u(c))/2$. This is the same point a standard parameterization based remesher would use. Note that in this case the detail vector is non-normal and its three components need to be stored.

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 $u(v)$ in the parameter domain. Let us denote by $u(q)$ the parametric coordinates of that piercing point.

We accept this point as a valid point of the semi-regular mesh if $||u(q) - u(v)|| < \kappa ||u(a) - u(v)||$, where $\kappa$ is an “aperture” parameter that specifies how much the parameter value of a pierced
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 \( u(\nu) \), resulting in a non-normal detail.

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 \( u(\nu) =: u(q) \). 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 \( u(q) \) in the parameter domain. This gives a new curve on the surface which passes through \( q \), 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 \( q \) 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 [16], so that the new net of curves 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 [18]. An example of parameterization adjustment after two levels of adaptive subdivision is shown in the bottom middle of Figure 5. Note that as the regions for which we compute parameterizations become smaller, the starting guesses are better and the solver convergence becomes faster and faster.

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

The aperture parameter \( \kappa \) of the piercing procedure provides control over how much of the original parameterization is preserved in the final mesh and consequently, how many non-normal details will appear. At \( \kappa = 0 \) we build a non-normal 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.](image)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Base mesh size</th>
<th>Normal mesh size</th>
<th>Not normal (%)</th>
<th>( L^2 ) error (min)</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feline</td>
<td>49864</td>
<td>156</td>
<td>40346</td>
<td>729 (1.8%)</td>
<td>.015</td>
<td>4</td>
</tr>
<tr>
<td>Molecule</td>
<td>10028</td>
<td>37</td>
<td>9521</td>
<td>270 (2.8%)</td>
<td>.075</td>
<td>1.5</td>
</tr>
<tr>
<td>Rabbit</td>
<td>16760</td>
<td>33</td>
<td>8235</td>
<td>196 (2.4%)</td>
<td>.037</td>
<td>2</td>
</tr>
<tr>
<td>Torus3</td>
<td>58844</td>
<td>98</td>
<td>5294</td>
<td>421 (8.0%)</td>
<td>.03</td>
<td>3</td>
</tr>
<tr>
<td>Skull</td>
<td>20002</td>
<td>112</td>
<td>25376</td>
<td>817 (3.2%)</td>
<td>.02</td>
<td>2.5</td>
</tr>
<tr>
<td>Horse</td>
<td>48485</td>
<td>234</td>
<td>59319</td>
<td>644 (1.1%)</td>
<td>.004</td>
<td>6.8</td>
</tr>
</tbody>
</table>

Table 1: Summary of normal meshing for different models. The normal mesh is computed adaptively and contains roughly the same number of triangles as the original mesh. The relative \( L^2 \) errors are 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 the following: 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, feline, and skull, as well as close-up of feline (bottom left) normal mesh. 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” \( \kappa \) 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.

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 operations such as smoothing, enhancement, and denoising can be computed through a suitable scaling of wavelet coefficients [7]. In a normal mesh any such algorithm will require only $1/3$ as many computations. Also large scaling coefficients in a standard mesh will introduce large tangent 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 implemented 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 coefficients 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. It should be possible to represent a $M$ dimensional manifold in $N$ dimensions with $N - M$ variables as opposed to the usual $N$.

- The current implementation only works for surfaces without boundaries and does not deal with feature curves. We will address these issues in our future research.

Acknowledgments This work was supported in part by NSF (ACI-9624957, ACI-9721349, DMS-9874082, DMS 9872890), 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 Zoe Wood for helpful discussions, Chris Johnson for the use of the SGI-Utah Visual Supercomputing Center resources, and to Cici Koenig for production help. Datasets are courtesy Cyberware, Heads, The Scripps Research Institute, and University of Washington.

References

Figure 10: Colorplate.