Variational Normal Meshes

ILJA FRIEDEL
Caltech
and
ANDREI KHODAKOVSKY
NVIDIA
and
PETER SCHRÖDER
Caltech

Hierarchical representations of surfaces have many advantages for digital geometry processing applications. *Normal meshes* are particularly attractive since their level to level displacements are in the local normal direction only. Consequently they only require scalar coefficients to specify. In this paper we propose a novel method to approximate a given mesh with a normal mesh. Instead of building an associated parameterization on the fly we assume a globally smooth parameterization at the beginning and cast the problem as one of perturbing this parameterization. Controlling the magnitude of this perturbation gives us explicit control over the range between fully constrained (only scalar coefficients) and unconstrained (3-vector coefficients) approximations. With the unconstrained problem giving the lowest approximation error we can thus characterize the error cost of normal meshes as a function of the number of non-normal offsets—we find a significant gain for little (error) cost. Because the normal mesh construction creates a *geometry driven* approximation we can replace the difficult geometric distance minimization problem with a much simpler least squares problem. This variational approach reduces magnitude *and* structure (aliasing) of the error further. Our method separates the parameterization construction into an initial setup followed only by subsequent perturbations, giving us an algorithm which is far simpler to implement, more robust, and significantly faster.

Categories and Subject Descriptors: G.1.2 [**Numerical Analysis**]: Approximation —*Approximation of surfaces and contours*; G.1.2 [**Numerical Analysis**]: Approximation —*Wavelets and fractals*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

General Terms: Surface approximation.

Additional Key Words and Phrases: (semi-)regular meshes, subdivision, normal meshes, hierarchy, resampling.

1. INTRODUCTION

One of the fundamental questions of surface representation concerns the relation between approximation quality and size of the representation. Even though a full theoretical characterization is not yet available, the practical importance of efficient representations for digital geometry processing is so great that a broad variety of algorithms have been put forward. Of particular interest in the context of display, editing and compression applications are *multiresolution* representations based on irregular [Hoppe 1996] and (semi-) regular meshes [Zorin et al. 1997; Gu et al. 2002]. The latter have many connections with classical functional representations such as wavelets [Schröder and Sweldens 1996] and Laplacian pyramids [Burt and Adelson 1983], which can be leveraged for digital geometry processing applications [Schröder and Sweldens 2001].

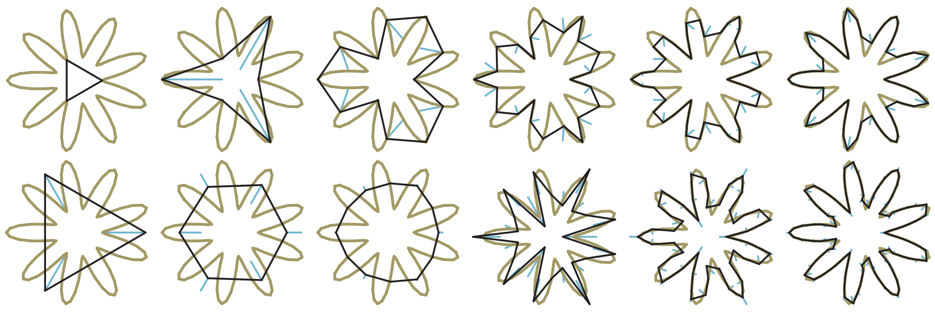


Fig. 1. The approximation errors of *interpolating* normal curves (top) are typically larger than for their variational counterparts (bottom). Note how the latter are low pass approximations until there are enough vertices to resolve the radial frequency avoiding aliasing artifacts (top). All coefficients (light blue) are scalar.

Of the (semi-)regular surface representations *normal meshes* [Guskov et al. 2000]—and their non-hierarchical relatives, *displaced subdivision surfaces* [Lee et al. 2000]—are of particular interest. Normal meshes are a hierarchical representation in which almost all coefficients are scalar rather than 3-vector valued. That is, level l is given as an offset from the coarser level $l-1$, with each offset being along the local normal direction on the surface. This immediate reduction in size by a factor of three can be exploited, *e.g.*, in compression applications [Khodakovsky and Guskov 2002; Lavu et al. 2003] and the representation of displacement maps [Lee et al. 2000]. Unfortunately only few theoretical results, which could guide the construction of normal meshes, are known so far. For example, in [Daubechies et al. 2004] it was shown that normal curve parameterizations possess (essentially) the same smoothness as the underlying coarse to fine predictor. The bivariate *functional* setting was studied in [Jansen et al. 2003] for purposes of compression.

One expects that the best results in terms of minimizing approximation error can be achieved without any constraints on the hierarchical displacement vectors. What is the penalty in terms of error if one insists on normal displacements only? What is the trade-off between allowing some non-normal coefficients and associated reduction in error? In this paper we explore these questions and will provide an algorithm that provides explicit control over this tradeoff.

To gain the advantages of normal mesh representations arbitrary input geometry must be remeshed so that almost all offsets are in the normal direction only. Guskov and co-workers [2000] formulated this as a resampling problem using a recursive triangle quadrisection procedure based on smooth interpolating subdivision [Zorin et al. 1996]. All vertices produced by this process are samples of the original surface. Since no low pass filtering is performed this leads to aliasing artifacts (see Figure 1). Constraining all vertices to lie on the original mesh also increases the approximation error vis-a-vis methods which allow a more unconstrained placement of vertices. In the method of Guskov *et al.* the parameterization needed for resampling was computed on the fly, a process which is rather expensive and numerically very delicate, in particular for large meshes.

1.1 Contributions

Our goal is the construction of low error approximations of a given surface with a (semi-) regular mesh while minimizing the number of non-normal coefficients. We control this trade-off by controlling the *perturbation of an initial, globally smooth parameterization* during the normal mesh construction process. This is in contrast to previous methods which computed a parameterization on the fly. The perturbation creates an explicit association between the original and approximating surface, which is driven by the *geometry*. Consequently it becomes meaningful to ask for the best approximation in the *mean squared distance* sense. A simple variational problem, to be solved at each level of the hierarchy, results in an approximation which is least squares optimal *for all levels* subject to a constraint on the magnitude of the parameterization perturbation. We achieve an overall reduction in error *and* better control of aliasing—the variational normal mesh is *approximating* not interpolating anymore (see Figure 1). The trade-off between normality and least squares optimality can be controlled explicitly and we show in the results section that the penalty—increase in approximation error—is small compared to the gain—reduction from 3-vector coefficients to scalars.

As a bonus, the separation of the (upfront) construction of a globally smooth parameterization—for which a variety of methods are available—from the rest of the algorithm, *greatly* simplifies the implementation, increases numerical robustness, and leads to significant speedup in total processing time.

Note: In the following sections we will use the curve case to illustrate all the basic ideas before giving details of the final algorithm for the mesh case.

2. BACKGROUND

To set the stage we briefly recall the salient features of the construction of [Guskov et al. 2000] before describing the reasoning behind the variational approach.

2.1 Interpolating Normal Meshes

In [Guskov et al. 2000] it was observed that one can construct curves in the plane and surfaces in 3D by specifying a hierarchy of mostly scalar offsets for the mesh vertices. In the construction of normal curves one starts from a polyline S^0 that interpolates the reference curve R. Each segment of S^0 is divided into two smaller segments by inserting a point p, using, *e.g.*, the midpoint rule. A detail coefficient is constructed by shooting a ray from p in the normal direction n of S at p (see Figure 2). The ray intersects the reference R one or more times. To avoid flips only intersections parametrically *between* the endpoints of the base segment are considered. One of the intersections r is picked by some heuristic—the algorithm works for a range of choices—and the scalar normal offset t is computed using $r = p + t \cdot n$. Sometimes r corresponds to a parametric location on R which is "far" from the parametric midpoint. For example, for reasons of avoiding too high a distortion one may want to reject locations r which are very close to one of the endpoints of R. In this case a vectorial offset ("non-normal coefficient") from p to the parametric midpoint of R is chosen. This decision process is typically controlled by an "aperture" around the parametric midpoint. This finishes the construction of S^1 and the process can now be repeated to obtain further refinements (Figure 2).

The algorithm was extended to surfaces by drawing curves onto irregular meshes [Guskov et al. 2000]. The surface was pierced by rays as described before in 2d. But because curves on manifolds are not necessarily flat the rays would pierce the surface at some distance from the existing curve network. To keep the initial parameterization consistent with the parameter values of the newly inserted points, the method recomputed the parameterization obtained in the previous level and redrew the curves *through* the new intersection points. This effectively meant that after each level of refinement the entire surface parameteriza-

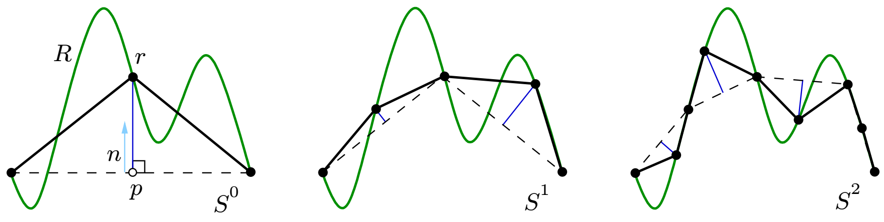


Fig. 2. Three levels of interpolating normal curves construction.

tion had to be recomputed. Not only is this costly: In general the irregular mesh and the (semi-)regular curve network overlap arbitrarly. The enforcement of the intersection constraints during relaxation also has the potential of cutting triangles at poor aspect ratios. This made the original method numerically challenging.

To overcome these difficulties we make two observations that will be important for us:

(1) The result of the naïve piercing algorithm [Guskov et al. 2000], which converges under mild technical conditions [Daubechies et al. 2004], depends only on the geometries of R and S^0. Any decisions to interfere with this process—treating an intersection as "too far off the middle"—are based on the ability to measure distances and find midpoints in the parametric domain Ω of R.

(2) If the interpolating normal curve refinement converges for inputs S^0 and R, then a reparameterization p^∞ is naturally defined by $S^\infty(t) = R(p^\infty(t))$ for all $t \in \Omega$. This p^∞ can be approximated on each level by a piecewise linear p^l such that for all vertices s_i of S^l we have interpolation $S^l(t_i) = s_i = R(u_i)$. Because s_i is attached to R at parameter value u_i we can construct $p^l(t_i) = u_i$. Now $S^l(t) \approx R(p^l(t))$—*implying that the difference between the two functions is a good approximation of their geometric distance.*

The latter observation is the starting point for our variational approach.

2.2 The Variational Approach

Given a parameterized curve or surface R and an approximation S^{l-1} on level $l-1$ we are interested in finding the coefficients of a refined approximation S^l such that distance decreases: $d(R, S^l) < d(R, S^{l-1})$. Ideally this distance should be measured using the symmetric Hausdorff metric [Cignoni et al. 1998]. Unfortunately this is costly, leading to the common use of the L_2 norm of the distance function

$$\|d_R\| := \Big(\int_\Omega (d_R(\omega))^2\, d\omega\Big)^{\frac{1}{2}}$$

as a way to evaluate the approximation error. Here $d_R(\omega)$ is defined on S and gives the distance to the nearest point on R.

Since a parameterization of the surface gives a functional description of the surface, an even simpler norm involves parameterizations of either surface

$$\|R - S\| := \Big(\int_\Omega (R(\omega) - S(\omega))^2\, d\omega\Big)^{\frac{1}{2}}. \tag{1}$$

This expression, unlike the L_2 norm of the distance function depends on the parameterization chosen for R and S. To make it geometrically meaningful, one needs to ensure that similar parameter values describe similar regions of R and S. This can be achieved by carefully selecting a suitable reparameterization $p : \Omega \to \Omega$ for one of the surfaces. The main insight of our work is that this type of reparameterization is precisely what the "piercing" procedure in the normal mesh construction produces. Using $\|R \circ p - S\|$ as a distance measure one can then hope for a behavior that resembles the L_2 norm measure of the distance function. A consequence of using $\|R \circ p - S\|$ is that one can easily solve the variational problem

$$\arg\min \|R \circ p - (S^{l-1} + \sum_i c_i^l \phi_i^l)\|^2 \tag{2}$$

to obtain detail vectors c_i^l describing S^l relative to S^{l-1}. The ϕ_i^l are the basis functions of S^l—piecewise linear hats in the case of meshes. The critical advantage of Eq. (2) is that it defines a positive semidefinite quadratic form. Finding optimal detail vectors $c_i^l \in \mathbb{R}^3$ (or $\mathbb{R}^2$ for curves) requires only the solution of a linear system. Note that we have not yet restricted the c_i^l to be scalar.

Repeating this process at each level of refinement results in a hierarchy of coefficients c_i^l giving the best L_2 approximation *at each level*. For surfaces these coefficients can be arranged in a Laplacian pyramid [Burt and Adelson 1983]. Letting N be the number of coefficients in the finest level L, the total number of pyramid coefficients is $(1 + \frac{1}{4} + \frac{1}{16} + \dots)N \leq 4/3N$, a modest overhead for the flexibility afforded. An *orthogonal* wavelet hierarchy could reduce this to N coefficients; to our knowledge no such construction is available for general surfaces.

3. VARIATIONAL NORMAL CURVES

To turn the above ideas into a practical algorithm we need to make some specific choices:

—*scalar* detail coefficients are allowed for *odd* (new) and *even* (old) vertices anywhere in the hierarchy

—*vectorial* details are only allowed for odd vertices and will be used sparingly

—no flags, except whether an odd coefficient is scalar or vectorial, are created.

(The last choice is motivated largely by limiting the side information needed to inverse transform the hierarchical surface representation.) In the standard interpolatory construction normal directions are used only once when moving a newly created (odd) vertex to its position on the reference curve R. In the variational algorithm we need to keep directions fixed, but allow vertices to slide along their *normal line*. A normal line corresponding to a vertex s_i of S is defined by its position and normal vector *at insertion time*. Vertices are free to slide along their normal lines, but are never allowed to leave them. We must allow such motion to ensure that the vertex s_i can converge for $l \to \infty$ to the intersection point of its normal line with R. Directions of normal lines though are held fixed once they have been created.

The variational refinement algorithm for curves consists of the following steps:

(1) refine mesh S^{l-1} by predicting odd points;

(2) find intersections of predicted normal lines with R;

(3) accept an intersection or select a vectorial offset;

(4) update the parameter perturbation p^l from p^{l-1}

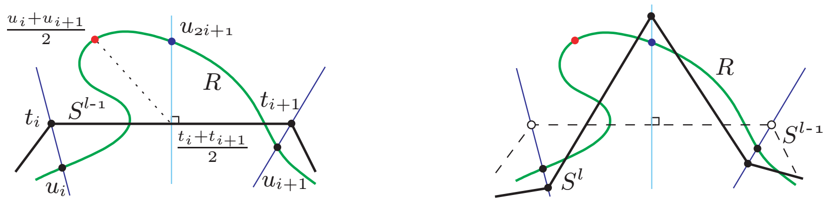


Fig. 3. Construction of approximating normal curves: correspondence of parameter values (left) and position of verices after minimization (right).

(5) define (tangentially displaced) normal lines for vectorial offsets;

(6) minimize the variational functional restricted to normal lines to obtain coefficients describing S^l.

The first three steps are essentially the same as in the interpolating curve construction. Here we focus on the remaining steps.

—The perturbation p^l is constructed by keeping $p^l(t_{2i}) := p^{l-1}(t_i)$ for even vertices (see Figure 3 for the various parameter locations and values). Let $R(u_{2i+1})$ be the intersection of the normal line with the reference surface and set $p^l(t_{2i+1}) := u_{2i+1}$ (blue dot on R). For a non-normal coefficient, inserted at the parametric midpoint of R, we would use a parameter value of $(u_{2i} + u_{2i+2})/2$ (red dot on R). This is all we need to define the piecewise linear reparameterization on the new level. Note that once a parameter value u is associated with t through the perturbation p it will never change.

—Having defined the new parameterization, Eq. (2) is well defined at level l and we may minimize it to determine the c_i^l. In Figure 3 the coefficients c_i^l move vertices along the normal lines, but in general do not interpolate R.

—Non-normal offsets should be allowed to participate in the minimization scheme. For this purpose we assign such coefficients a (translated) normal line anchored at $R(u_{2i+1})$, parallel to the originally predicted normal direction n_{2i+1} of S. Instead of recording the vectorial offset to $R(u_{2i+1})$ plus the scalar coefficient c_{2i+1} resulting from the minimization, we only record the final positions $R(u_{2i+1}) + c_{2i+1} \cdot n_{2i+1}$ of these vertices and use these as the origin of the associated normal lines.

Computing the minimum of a quadratic form requires the solution of a linear system $b = K \cdot c$ of normal equations: $b_i = \langle R, \phi_i^l \rangle$ defines the load vector and $K_{ij} = \langle \phi_i^l, \phi_j^l \rangle$ the mass matrix. If no area weighting is used the entries of K can be computed offline. In practice though it is more appropriate to take the actual triangle sizes into account. Computation of the entries of b requires online numerical quadrature because of their dependence on R. In any case, the setup of the linear system is straightforward.

Even though the approximation is not interpolating we are using the fact that the *basis functions* are interpolating. Consider two neighboring basis functions ϕ_i, ϕ_j and suppose that ϕ_i is nonzero at the normal line of ϕ_j. Changing the coefficient of c_i could then "push" c_j off its normal line unless the two normal lines happen to be parallel. Because interpolating basis functions, such as piecewise linear hats, are zero at the normal lines of all other vertices we do not need to worry about this effect.

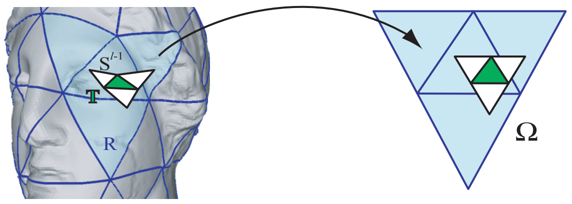


Fig. 4. Flattening of a region around remesh triangle T defined by a base patch and its three neighbors.

4. VARIATIONAL NORMAL MESHES

As noted before we do not compute a parameterization of the mesh on the fly but rather rely on a pre-existing parameterization. It could be produced with any of the algorithms now available for the construction of low distortion, globally smooth parameterizations. Our only desirable is that the parameterization be *locally* close to an isometry to simplify finding reasonable “midpoints” between two vertices in the non-normal case.

The parameterization could be given on the original unstructured mesh or in the form of a (semi-)regular remesh. We prefer (semi-)regular parameterizations, *e.g.*, those produced by MAPS [Lee et al. 1998], GI [Gu et al. 2002], or GSP [Khodakovsky et al. 2003] for practical reasons. The operations needed (finding midpoints and computing distances in the parameter domain) are well supported by (semi-)regular meshes:

—Evaluating a surface R for a given base patch of S at some barycentric coordinate is easily realized through a logarithmic time hierarchy traversal.

—The inverse operation, *e.g.*, turning a ray intersection at some fine level into a coordinate value with respect to the base domain, is similarly easy to implement and efficient to run.

This allows the computation of parametric distances *within* a base patch. Using a (semi-) regular parameterization also reduces the complexity of flattening R locally, which is needed if distances are to be computed *across* base patch boundaries.

In terms of the above assumption on the input our algorithm starts with a hierarchy of meshes $R^0, R^1, \ldots, R^L = R$. With $S^0 := R^0$ as the base domain the parameterization perturbation starts with the identity, $p^0 := id$. Note that if vertex insertion were always performed at parametric midpoints of R, all offsets would (in general) be vectorial and for all i, $p^i := id$.

Let T be some triangle of the normal remesh S^{l-1}. This triangle (green in Figure 4) and its neighbors (white) are in most cases completely contained inside a base domain triangle (blue boundaries). In this case one can compute midpoints and distances within the parametric domain as described. (For a remesh triangle that is not completely contained within a single base domain patch see below.)

Once we have flattened R in a neighborhood of T we can make decisions on the piercing points. The triangle T and its three neighbors (see Figure 5 which shows the parametric domain) are associated with R via p^{l-1}. The piercing procedure begins by shooting rays from the midpoints of the edges in the normal direction to S^{l-1}. The normal direction at the midpoint of an edge is set to bisect the dihedral angle of the two incident triangles. These rays will generate intersections with R (otherwise the distance to the intersection is set to ∞ and a non-normal offset is created). Given the current parameterization these

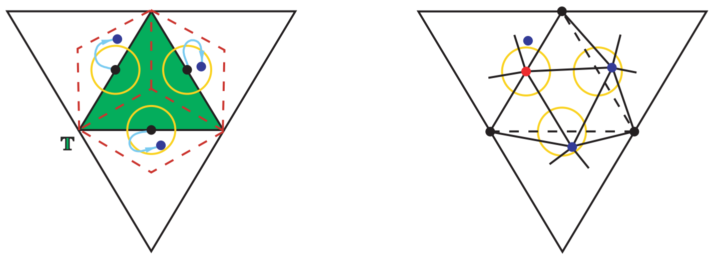


Fig. 5. Piercing and reparameterization in the parameter domain: One of the new details (blue) pierces the mesh outside of its apperture (yellow circles). This causes the creation of a non-normal vertex, whose parameter value (red dot) remains as predicted. The parameter values of normal vertices on the other hand are slightly perturbed.

intersection points correspond to blue dots in Figure 5 (left). If these intersections lie within their respective red diamonds we can guarantee that no folds in the parameterization will occur. Otherwise we reject the intersection. In fact we limit legal intersections to the yellow aperture for simplicity: a radius/distance bound is easier to check than containment in a diamond.

As mentioned before we sometimes need to compute distances across base domains as shown in Figure 4. In this case we attempt to create a larger, flat domain of R that includes T and its neighbors. In [Khodakovsky et al. 2003] exactly this problem was solved (iteratively) by expressing the barycentric coordinates of one base domain triangle with respect to a selected neighboring base domain triangle. Doing so is somewhat involved, also because one would need to select a specific sequence of domain crossings.

We avoid the problem of selecting this sequence of crossings by performing only one step of the process, *e.g.*, by flattening the three neighbors of a base domain triangle only (Figure 4). This is done using the *hinge map* of [Lee et al. 1998; Khodakovsky et al. 2003], which simply extends the barycentric coordinates of a triangle to its three neighbors. In the very rare case that an even larger flattened domain is needed, the algorithm creates a non-normal vertex. We have not observed any negative impact of this restriction in our experiments. (Larger parametric displacements are rare and in any event are better dealt with through a non-normal coefficient.) Thus the worst case requires flattening a base mesh triangle patch of R and its three patch neighbors.

Now associate the new vertices of S^l with the parametric values of the intersections, as in the curve case, to build the new piecewise linear p^l. Because the topology of p^l is the same as of S^l one does not need to construct a new mesh for p^l. Instead we store the parameter values as attributes of the vertices in S^l. Figure 5 (right) shows the new S^l with one intersection rejected and replaced with a point on R which corresponds to the parametric midpoint (red dot) analogous to the curve case.

To solve for the c_i^l, *i.e.*, the final location of the vertices of S^l along their normal lines, we need to set up the least squares system. For hat functions, without taking account of the surface element on R, the mass matrix has entries $K_{ii} = valence_i/12$ and $K_{ij} = 1/12$ if i and j are connected by an edge. This matrix, for example, was used in [Lounsbery et al. 1997] for the construction of wavelets over (semi-)regular meshes. Since triangles are generally not uniform in size we use numerical integration to compute the entries of K and take the actual surface area into account. For this we employ the midpoint quadrature rule with between 30 and 150 samples per triangle of S^l.

This concludes the description of the algorithm.

5. IMPLEMENTATION AND RESULTS

Most of the components needed for the implementation of a variational normal remesher: mesh library, ray-surface intersection, and linear solver, were taken off the shelf. The only custom implementation was the code for flattening of base triangles of R. For the variational normal mesh (VNM) code a simple numerical integrator (midpoint) was added. We did not explore the trade-offs due to numerical integration accuracy and final approximation error (we use between 30 and 150 integration points per triangle).

For both INM and VNM the *observed* runtime is linear in the number of triangles. (Note that while individual point locations are $O(\log n)$ their expected cost is $O(1)$ explaining the observed behavior.) The runtime of the VNM remesher is completely dominated by the integration code (see the representative data in Table I). The timing differences between INM and VNM are due to linear equation system setup and solution. The linear solver time is on the order of a second hence the difference is in essence the cost of integration. The fact that the INM code is now so fast is partially due to the simpler flattening procedure, but also to having replaced the on the fly repeated reparameterization [Guskov et al. 2000] with an upfront parameterization. Even for the VNM though our results compare favorably with Guskov *et al.* (accounting for our timings being taken on a 2.2 GHz P4). The initial parameterization was essentially free as we relied on available remeshes [Lee et al. 1998; Khodakovsky et al. 2003]. Some models are not readily available as remeshes. Here one has to take the parameterization time into account. Remeshing algorithms have evolved significantly over the past few years (see for instance [Lee et al. 1998; Khodakovsky et al. 2003; Aksoylu et al. ; Schreiner et al.] for timings). The best results so far where obtained by [Aksoylu et al.] who report solver timings of under 40 seconds for a model containing 580k vertices (David head).

We have run experiments with a range of MAPS and GSP input parameterizations. The remeshing errors of our INM algorithm are about the same as in [Guskov et al. 2000] (see Table I for our results).

The anti-aliasing properties of variational normal meshes are clearly visible in the “zone” sphere example of Figure 6.

In terms of error, VNM give us a fairly consistent improvements over INM. Typically INM have up to 60% larger remeshing error (on any level) relative to VNM. Figure 7 shows comparisons between different normal mesh types for different models and the GSP input parameterization. In particular we compare against the vectorial variational mesh (VVN), where detail vectors are not direction constrained. All errors where computed with METRO [Cignoni et al. 1998]. For the feline and igea models we compared against the *original, irregular meshes* from which the GSP where derived; while the dino and zone-sphere models are compared against a finer, (semi-)regular mesh. The only difference observed is the GSP remeshing error on the finest level of feline and igea graphs.

We observe, that both interpolating methods (INM, GSP) and also both approximating methods (VNM and VVN) perform roughly the same. Variational meshes (VNM and VVM) also preserve volumes equaly well - much better than the interpolating hierarchies (INM, GSP). This behavior is illustrated by the skull series in Figure 8 and the error graphs in Figure 9.

The number of non-normal coefficients we achieved is typically a little less (we are

data set	input param	normal method	input base mesh size (♯ levels)	remesh size (vertices)	non-normal vertices	percent B-box L_2 Error	Time (sec)
skull	MAPS	INM	4(8)	32770	368	0.0392	2.5
	MAPS	VNM	4(8)	32770	494	0.0282	15.2
fandisk	MAPS	INM	73(4)	4546	103	0.0573*	0.2
	MAPS	VNM	73(4)	4546	104	0.0345*	1.5
dino	MAPS	INM	128(4)	8066	228	0.0893*	0.3
	MAPS	VNM	128(4)	8066	294	0.0576*	2.5
igea	MAPS	INM	196(5)	49666	136	0.0148	2.3
	MAPS	VNM	196(5)	49666	121	0.0096	14.9
	GSP	INM	40(6)	38914	24	0.0156	2.1
	GSP	VNM	40(6)	38914	38	0.0099	15.1
feline	GSP	INM	280(5)	72190	589	0.0156	3.4
	GSP	VNM	280(5)	72190	845	0.0096	25.3
horse	GSP	INM	140(5)	35330	256	0.0117	1.6
	GSP	VNM	140(5)	35330	317	0.0081	11.7
rabbit	GSP	INM	100(5)	25090	20	0.0107	1.1
	GSP	VNM	100(5)	25090	24	0.0067	8.6
zone-	Loop	INM	12(7)	40962	570	0.0611	2.8
sphere	Loop	VNM	12(7)	40962	146	0.0327	17.2

Table I. Using MAPS parameterizations as input to our algorithm gives us similar remeshing errors as when using GSP. But typically the number of non-normal vertices is higher for MAPS, reflecting the fact that MAPS parameterizations are not globally smooth. Variational normal meshes (VNM) typically outperform their interpolating (INM) counterparts. Errors where computed using METRO with respect to the original, irregular mesh. An exception are the fandisk and dino models, which where compared against the finest level MAPS remesh. Hence the MAPS remeshing errors need to be added to these numbers. (We discovered that the MAPS remeshes are scaled/rotated versions of the irregular models publicly available.)

using a constant aperture of 0.2 for all levels while in [Guskov et al. 2000] the aperture was relaxed from 0.2 on coarse to 0.6 on finer levels). The variability in these numbers is not suprising, because the construction of normal meshes depends on base mesh and the parameterization chosen for the metric.

As in the original paper we have used a spatially invariant aperture to remesh from one level to the next. This works well in regions with simple geometry and “nice” input parameterization. In those settings no non-normal coefficients are inserted (see the feline trunk in Figure 10). In regions of high curvature non-normal coefficients *are* inserted, preventing mesh degeneration. Interestingly, flat regions sometimes produce non-normal coefficients due to excessive distortion in the original input parameterization (see the feline wing attachment and tips). Increasing the aperture locally one can get rid of this problem and a nice reparameterization results (see Figure 11).

The VNM algorithm samples the geometry of the input mesh fairly densely (as part of the integration routine). Thus one could hope to find a strategy that adopts the aperture locally based on this information at no extra cost. We did not run experiments to examine such strategies.

6. CONCLUSION AND FUTURE DIRECTIONS

For many digital geometry processing applications, *normal meshes* are an attractive (semi-) regular, hierarchical representation. Hitherto their construction was delicate and compu-

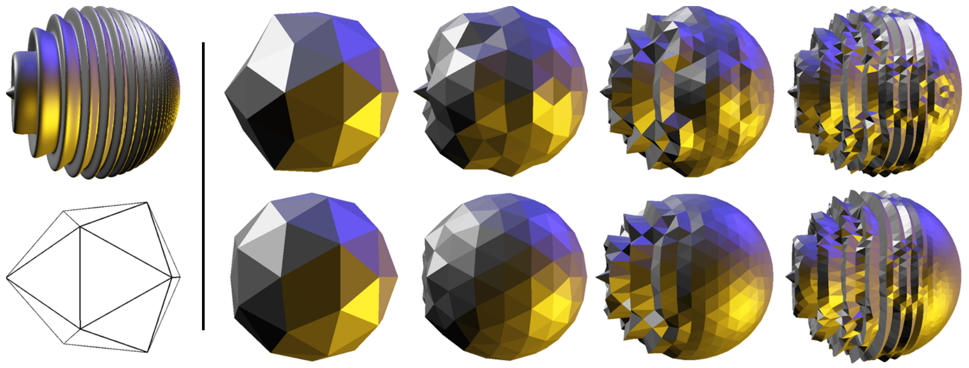


Fig. 6. A fine sampling of a “zone” sphere with a displacement field of increasing frequency (moving along the equator) is used to test for aliasing properties (leftmost image, also showing the icosahedral base mesh). On the right the upper row shows levels 1 to 4 of the interpolating normal mesh refinment. The right hemispheres, which contain high frequencies in the original geometry, exhibit aliasing artifacts in the interpolating construction. The corresponding variational normal meshes (bottom row) correctly low pass filter frequencies which cannot be represented at the current resolution.

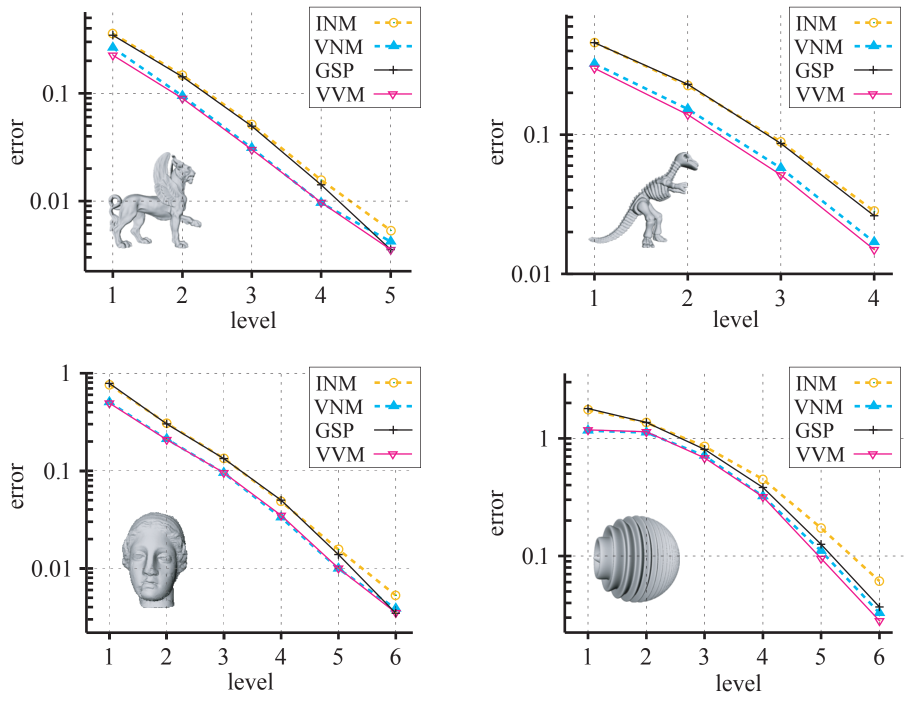


Fig. 7. The METRO mean squared distance errors (percent B-box) are plotted for four different models using the GSP input, normal remeshes which are: interpolating (INM), variational (VNM); and also for unconstrained variational solutions (VVM). These examples illustrates how close the constrained variational normal meshes are to the unconstrained variational meshes. Note that for the feline and igea models the errors are measured with respect to the original irregular triangle mesh, while the dino and the zone sphere meshes are compared against the finest level (semi-)regular mesh available.

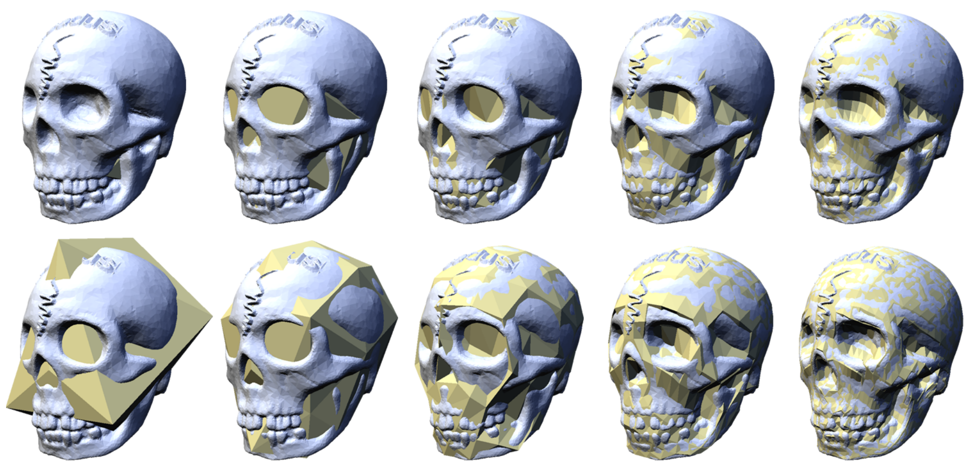
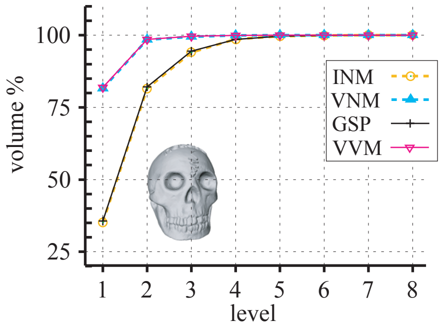


Fig. 8. Interpolating normal meshes are completely contained inside of convex regions of objects (top row, levels 1 to 5). This causes large errors for the volume of the reconstruction. Variational normal meshes place vertices at optimized positions (bottom row) and preserve the volumes better.



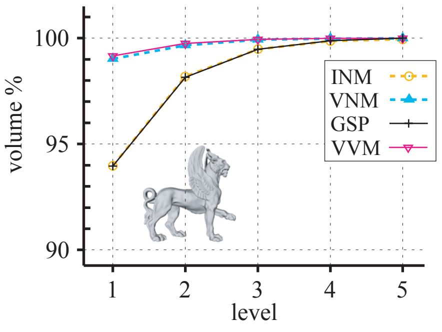


Fig. 9. These graphs are typical plots displaying the relative volumes (in percent of the original irregular mesh) of the 4 mesh hierarchies. Variational meshes consistently preserve volumes better than interpolating hierarchies. Still they are still slightly biased towards underestimating the true volume. The skull base mesh is a tetrahedon, hence the graphs show much larger volume defect than other meshes with more detailed base meshes. The relative behavior of interpolating to variational errors nevertheless is very similar.

tationally expensive and the impact of the coefficient normality constraint on the approximation error was unclear. We find that our novel *variational normal mesh* approach yields very high approximation quality: with only a small number of non-normal coefficients we achieve error essentially as good as that produced for meshes with the full freedom of 3-vector displacements everywhere. Variational normal meshes define their coefficients through a squared distance minimization problem which closely mimics the geometric distance minimization, but is far simpler to solve. The resulting meshes are *approximating*, which helps us reduce the magnitude of the error and the aliasing artifacts present in interpolating constructions. The resulting algorithm is far simpler, more robust and faster than the original method.

The basis functions in our construction are still interpolating (though not the approx-

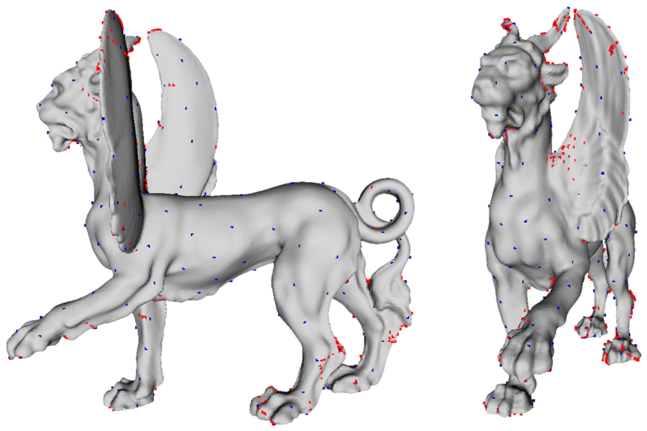


Fig. 10. An interpolating normal mesh (INM) of the feline dataset. Vertices of the base mesh S^0 are shown in blue while non-normal displacements (relative aperture size of 0.2) are colored red. Most non-normal displacements are due to severe geometric distortion (paws, edge of wing, *etc.*). However, there also some non-normal coefficients in geometrically "flat" regions. These are due to parametric distortion (see Fig. 11) causing essentially tangential displacements. The location of non-normal coefficients for VNM are very similar for this geometry.

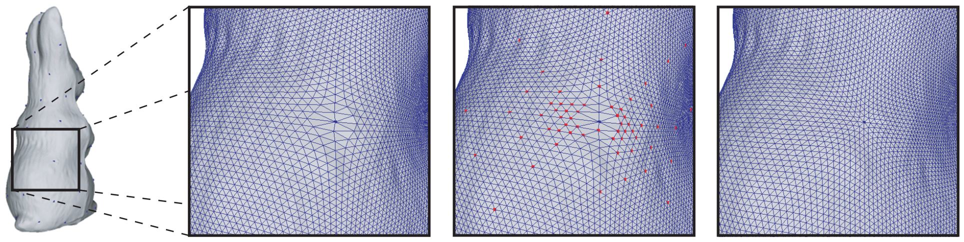


Fig. 11. A closeup of the neighborhood of a base mesh vertex (blue) of high valence. The distortion in the input parameterization is clearly visible (left). Because the geometry is simple, a nice remesh is achieved if we do not interfere with the normal remesh (aperture 0.3, right box). A small aperture (0.05) allows for only a small perturbation of the input mesh (middle) and results in more non-normal coefficients due to tangential displacement (red dots).

imation itself). Perhaps even better approximations can be built when using, *e.g.*, cubic B-splines. In that case all coefficients would have to be solved for simultaneously in a *non-linear* minimization problem. Another interesting avenue for future work is the construction of normal meshes fine to coarse. These offer the potential for fast transform methods.

ACKNOWLEDGMENTS

This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), the DOE (W-7405-ENG-48/B341492), NVIDIA, the Center for Integrated Multiscale Modeling and Simulation, Alias|Wavefront, and Pixar. The models used in this paper are courtesy of Cyberware (rabbit, dino, skull, igea), Georgia Tech Large Geometric Model Archive (horse) and Hughes Hoppe (fandisk). Special thanks to Mathieu Desbrun, Nathan Litke, Sharif Elcott and Dave Felt.

REFERENCES

AKSOYLU, B., KHODAKOVSKY, A., AND SCHRÖDER, P. Multilevel solvers for unstructured surface meshes. accepted, SIAM J. Sci. Comput.

BURT, P. J. AND ADELSON, E. H. 1983. The Laplacian Pyramid as a compact image code. *IEEE Transactions on Communications 31*, 532–540.

CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. 1998. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum 17,* 2, 167–174.

DAUBECHIES, I., RUNBORG, O., AND SWELDENS, W. 2004. Normal Multiresolution Approximation of Curves. *Constructive Approximation*.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry Images. *ACM Transactions on Graphics 21,* 3, 355–361.

GUSKOV, I., VIDIMČE, K., SWELDENS, W., AND SCHRÖDER, P. 2000. Normal Meshes. *Proceedings of SIGGRAPH 2000*, 95–102.

HOPPE, H. 1996. Progressive Meshes. *Proceedings of SIGGRAPH 96*, 99–108.

JANSEN, M., BARANIUK, R., AND LAVU, S. 2003. Multiscale Approximation of Piecewise Smooth Two-Dimensional Functions using Normal Triangulated Meshes. *Submitted for publication.*.

KHODAKOVSKY, A. AND GUSKOV, I. 2002. Normal Mesh Compression. In *Geometric Modeling for Scientific Visualization*, G. Brunnett, B. Hamann, and H. Müller, Eds. Springer Verlag.

KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P. 2003. Globally Smooth Parameterizations With Low Distortion. *ACM Transactions on Graphics 22,* 3, 350–357.

LAVU, S., CHOI, H., AND BARANIUK, R. 2003. Geometry Compression of Normal Meshes using Rate-Distortion Algorithms. In *Proceedings of Symposium on Geometry Processing*. Eurographics/ACM SIGGRAPH, 52–61.

LEE, A., MORETON, H., AND HOPPE, H. 2000. Displaced Subdivision Surfaces. *Proceedings of ACM SIGGRAPH 2000*, 85–94.

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

LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. 1997. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *ACM Transactions on Graphics 16,* 1, 34–73.

SCHREINER, J., ASIRVATHAM, A., PRAUN, E., AND HOPPE, H. Inter-surface mapping. accepted, SIGGRAPH 2004.

SCHRÖDER, P. AND SWELDENS, W., Eds. 1996. *Wavelets in Computer Graphics*. Course Notes. ACM SIGGRAPH.

SCHRÖDER, P. AND SWELDENS, W., Eds. 2001. *Digital Geometry Processing*. Course Notes. ACM SIGGRAPH.

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

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