Hybrid Meshes: Multiresolution using regular and irregular refinement

Igor Guskov
Univ. of Michigan

Andrei Khodakovsky
Caltech

Peter Schröder
Caltech

Wim Sweldens
Bell Labs

ABSTRACT

A hybrid mesh is a multiresolution surface representation that combines advantages from regular and irregular meshes. Irregular operations allow a hybrid mesh to change topology throughout the hierarchy and approximate detailed features at multiple scales. A preponderance of regular refinements allows for efficient data-structures and processing algorithms. We provide a user driven procedure for creating a hybrid mesh from scanned geometry and present a progressive hybrid mesh compression algorithm.

Categories and Subject Descriptors

I.3.5 [**Computational Geometry and Object Modeling**]: Surface representations

Keywords

Compression Algorithms, Curves & Surfaces, Geometric Modeling, Level of Detail Algorithms, Polygonal Modeling, Remeshing

1. INTRODUCTION

Highly detailed and complex surfaces are most commonly described through meshes. Typical sources of such meshes range from CAD systems (e.g., STL files) to range sensing [26] and isosurface extraction [27]. Sizes can easily reach to millions, and even billions, of vertices. Processing such digital geometry requires efficient algorithms and powerful mathematical tools. The connectivity of the meshes plays a major role in the construction of appropriate algorithms. Typically we distinguish between *irregular* and *(semi-)regular* meshes. Irregular meshes have no restriction on the valence of vertices, while regular meshes are formed by starting from a coarse irregular base domain and applying recursive regular refinement, resulting in large uniform grid patches. Both types of meshes come with their own advantages and disadvantages.

Since there is no valence restriction, irregular meshes are more flexible than regular meshes; they can better resolve complex geometric features and accommodate topology changes. This comes at the cost of more complex algorithms for multiresolution [14], smoothing [7], compression [3], editing [23], etc.

SoCG'02, June 5-7, 2002, Barcelona, Spain.

Regular (or semi-regular) meshes have an almost everywhere regular structure, allowing for efficient tree based data structures with higher performance on modern processors. This regularity also enables the use of many traditional signal processing type algorithms based on Fourier [29], subdivision [34], and wavelet [21] methods. Because of their regular and hence predictable parametric structure and connectivity, regular meshes are inherently much more compressible than irregular ones [21].

The main drawback of regular meshes is their lack of flexibility in resolving shapes with high genus or features at many scales. A regular mesh always has a bijective mapping between the coarsest level (base domain) and the finest level, implying that the base domain has to have the same, possibly high, genus as the final model. In addition, to avoid excessive stretching of the map the complexity of the base domain must grow with the general complexity of the model. A stretched map can lead to bad aspect ratio polygons, poor approximation, and numerical problems. "Spiky" features typically lead to this phenomenon. Consider the bunny ears: unless the base domain has a few polygons outlining the ears, a semi-regular mesh will never be able to resolve the ears without very stretched polygons, see Figure 2 left. There must be enough "skin" in the base domain to avoid later stretching. Note that the stretching problem is fundamental to regular meshes. Adaptivity cannot help because regular refinement does not change the aspect ratio and reparameterization cannot help either as the curvature is inherent in the original model.

Both the genus and the stretching problem restrict how coarse the base domain can be and limit the scalability of semi-regular meshes. To address these problems the present paper introduces *hybrid meshes*, which combine advantages of regular and irregular meshes. Hybrid meshes use a preponderance of regular refinement inheriting their advantages, but also allow occassional irregular operations to grow extra "skin" or change topology within the hierarchy.

The Buddha model helps to illustrate the scalability of hybrid meshes: the base domain of a hybrid mesh is a few cubes, see Figure 1, while the semi-regular remesher of [15] results in an eight hundred polygon base mesh.

The contributions of this paper are the introduction of a multiresolution hybrid mesh representation for topologically and geometrically complex surfaces, the extension of signal processing tools and progressive compression for hybrid meshes, and a user driven remeshing procedure for constructing a hybrid mesh from given geometry.

Related Work. Changes in topology have been considered in the context of irregular hierarchy constructions through mesh simplification. For example, Garland and Heckbert [11] allowed for ver-

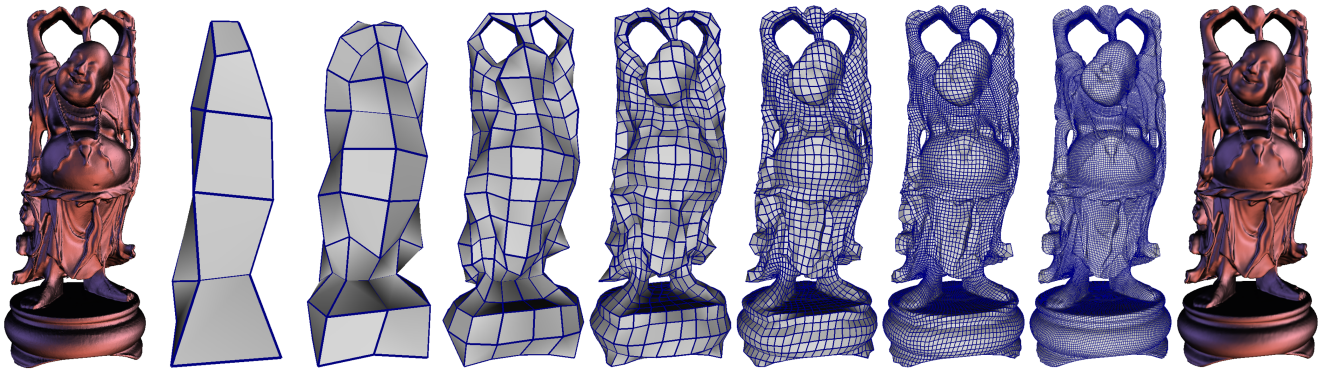


Figure 1: ***Example of a hybrid mesh (left to right). Starting with the Stanford Buddha original dataset a base domain with genus 0 is constructed. After one regular refinement step, tunnels are created between arms and head followed by another regular refinement step and creation of a tunnel between the feet. After one more refinement step three small tunnels are created on the sides followed by further regular refinement. The model remains a valid 2-manifold throughout and the final remesh has genus six.***

tex pair collapses during progressive mesh [17] construction, sacrificing the manifold property to better deal with many connected components. A fully general treatment was given by Popović and Hoppe [30]. Topology reduction as an explicit goal was pursued by El-Sana and Varshney [9]. None of these algorithms attempt to maintain the 2-manifold property as we do.

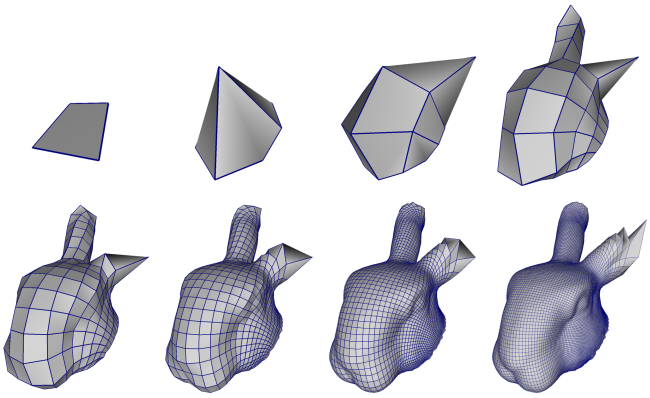


Figure 2: ***A remesh of the bunny head: the left ear uses a standard semi-regular mesh, while the right ear uses a hybrid mesh. Starting from a single quad extra "skin" is provided for the right ear after three regular refinement steps through the addition of three cubes. The resulting quads continue to have good aspect ratio. The left ear on the other hand suffers from severe stretching and even after seven refinement steps large parts of the left ear parameter space have not yet been sampled.***

In the ab initio modeling setting, the problem of stretched features and non-trivial topology was addressed by a number of researchers. For example, in Barghiel et al. [5] surface pasting is used to add features in a multilayered fashion. Changing topology while maintaining the manifold property in the context of a hierarchical modeler was first introduced by Gonzalez-Ochoa and Peters [13]. In a similar setting Akleman and Chen [1, 2] focus on the problem of maintaining the 2-manifold property for subdivision control meshes. While our hybrid mesh representation bears certain similarity to the surface representations used in ab initio modeling, in this paper we focus on the problem of constructing topology changing parameterizations for *given* geometry.

Most remeshing work has focused on the semi-regular setting, i.e., how to construct a semi-regular resampling of some irregular geometry. Hoppe and co-workers [18] used a global optimization framework to recover a Loop subdivision surface from a given point cloud. Their examples demonstrate how geometric detail increases the combinatorial complexity of the control mesh. In order to use surface based wavelet methods [28], which effectively add details to the subdivision setting, Eck and co-workers [8] gave the first automatic remeshing procedure, but with little control over the complexity of the base domain. In many settings a fully automatic method is not appropriate and patch layout under user control is essential. Krishnamurthy and Levoy [24] introduced such a system for the construction of bicubic patch layouts on scanned meshes, while Lee and co-workers [25] described a system with a flexible tradeoff between user control and automatic remeshing. These and other semi-regular remeshing algorithms all suffer from the same distortion issues when insisting on a coarse base domain and none can accommodate topology changes during remeshing.

2. HYBRID MESHES

We begin with the definition of hybrid meshes. We assume quadrilaterals as the basic face primitive and quadrisection as the associated regular refinement operation. Our definitions and algorithms would work equally well using triangles or hexagons with regular refinement, e.g., [22] and [33].

Hybrid meshes rely on the observation that most of the advantage of irregular meshes can be obtained by interspersing irregular operations in a preponderance of regular refinements. Thus one can think of a hybrid mesh is as a generalization of a semi-regular mesh in which irregularity is not constrained to the roots of the trees (Figure 3, left), but instead is allowed to occur distributed throughout the refinement hierarchy (Figure 3, right).

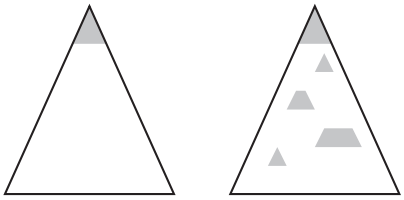


Figure 3: ***Semi-regular (left) vs. hybrid meshes (right): In a semi-regular mesh, only the top is irregular (shaded) and everything on finer levels is regular. In a hybrid mesh, irregular pieces can be distributed throughout the tree***

Formally, a hybrid mesh is a hierarchical mesh representation that starts with a coarsest base domain and builds finer levels using two types of operations:

- A *regular* operation in which a single quad (face) is split into four quads (regularly refined).
- An *irregular* operation in which a set of quads (faces) is replaced with another set of quads (faces) which agree on the boundary.

Figures 4 and 5 show a couple of simple examples of local irregular operations.

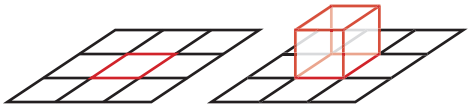


Figure 4: ***Add cube.***

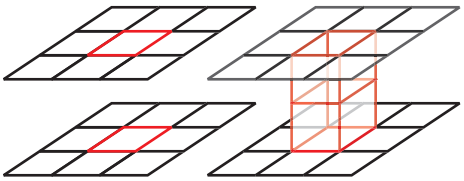


Figure 5: ***Connect.***

We introduce some notation for further use. Recall the structure of semi-regular meshes. They start with an irregular coarse base domain $\mathcal{M}^0$ and successively build finer levels $\mathcal{M}^j$ through regular refinement $\mathcal{R}$: so that $\mathcal{M}^j = \mathcal{R}\,\mathcal{M}^{j-1}$. For quads the number of elements grows by a factor of four from level $j-1$ to level j. Of course, in practice there is no need to instantiate refinements uniformly, allowing for adaptively refined meshes.

A hybrid mesh hierarchy retains the notion of levels: an irregular operation can only involve quads from the same level. A hybrid mesh hierarchy then consists of two types of meshes $\mathcal{M}^j$ and $\mathcal{N}^j$. The mesh $\mathcal{M}^j$ is built by regular refinement of mesh $\mathcal{N}^{j-1}$ ($\mathcal{M}^j = \mathcal{R}\mathcal{N}^{j-1}$), while the mesh $\mathcal{N}^j$ follows from performing one or more irregular operations on the mesh $\mathcal{M}^j$. The base domain is $\mathcal{M}^0 = \mathcal{N}^0$. Hence we have the sequence $\mathcal{N}^0 \rightarrow \mathcal{M}^1 \Rightarrow \mathcal{N}^1 \rightarrow \mathcal{M}^2 \Rightarrow \mathcal{N}^2 \ldots$, where $\rightarrow$ stands for regular refinement and $\Rightarrow$ for irregular operations within a level. A semi-regular mesh is now the special case where no irregular operations happen and all $\mathcal{N}^j = \mathcal{M}^j$. One can also imagine a hypothetical situation where the irregular operations affect all the quads in a level even on finer levels of the hierarchy. Then, none of the advantage of regular meshes would survive. Hybrid meshes need to use a majority of regular operations and limit irregular operations as needed for topology changes, feature alignment, and stretching resolution.

Hybrid data structures. Our hybrid mesh data structure is a collection of quad trees: we start out with one tree per quad of the base domain; every irregular operation terminates the subtrees formed by the quads that are removed and creates as many new trees as there are new quads. The only extra information that needs to be stored (as compared to the usual subdivision data structures) are the neighborhood relationships on the boundary of the repatched region to tie the trees together.

In the implementation we use two types of quads: root and non-root quads. Each root quad has four pointers to its neighbors and, if refined, has four pointers to its children. A non root quad has a parent pointer, no neighbor pointers, and, if refined, four pointers to its children. All regular refinement is performed uniformly, however, it may be *instantiated* lazily to provide the benefits of adaptive

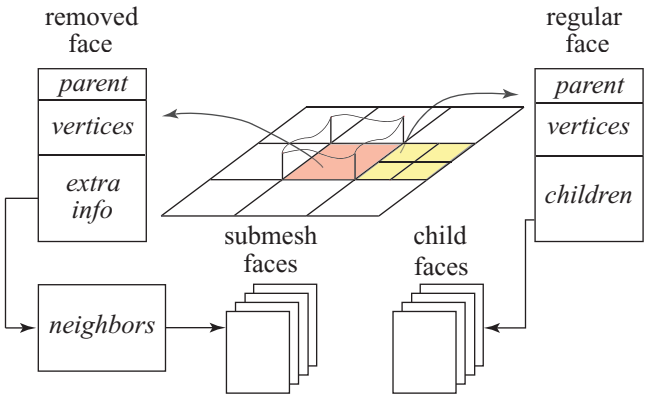


Figure 6: ***Face data structure. The face on the right is regularly refined. Its children slots point to actual children. The face on the left is marked for removal. Removed faces do not have children but the have pointers to neighboring faces.***

refinement. The latter is done with a standard restriction criterion. We do not allow irregular operations on root quads since root quads themselves already represent an irregular mesh.

Regular operations simply populate a standard tree. Irregular operations involve removing selected non-root quads as well as inserting new root quads in their stead. Putting in the neighbor pointers for the new root quads is straightforward. The only difference with respect to base domain root quads is that the neighbors need not be root quads themselves. When a non-root quad gets removed, it is replaced by a special data structure that stores the neighbor pointers of the removed quad to the newly created root quads. For example, for the add cube operation (Figure 4) we will store pointers to the four sides of the cube. In case the removed quad has no new root quad as a neighbor, there are no pointers to store. This happens for example when a single quad is removed to create a boundary, or in case a removed quad does not touch the boundary of the removed quad region.

There are standard algorithms for resolving neighborhood relations in quadtrees (e.g., [31]). Neighbor questions ascend up the tree, reach the nearest common ancestor, and descend back down to find the answer. For semi-regular meshes, one modification is needed: When the question reaches a root quad, its explicit neighborhood relationships are used before descending down the (neighboring) tree. For hybrid meshes, another modification is needed. When descending the tree, one can reach a removed quad. Then the explicit neighbor relations stored on the boundary between the regular region and irregular patch are used to continue the algorithm.

3. BUILDING HYBRID MESHES

In this section we describe our algorithm to build hybrid meshes. The input is a fine irregular triangle mesh. The system provides a mixture of automatic operations and interactive user control. Certain tasks such as tracing curves and smoothing patch boundaries are best done by the system while others such as deciding topological cuts or aligning patch boundaries with features are best done with user intervention [24].

In this section we first describe hybrid parameterization, then give a simple example, and finally outline topological operations.

3.1 Hybrid parameterization

As in semi-regular remeshing, parameterizations are the key to hybrid remeshing. In the semi-regular case, we find a single base or

parameter domain $\mathcal{B}$ and compute a parameterization of the original mesh $\mathcal{K}$ into the base domain. The parameterization is then use to resample and build finer semi-regular meshes [8, 25]. Hybrid remeshes differ from this, in that the parameter domain, and hence the parameterization can change as a result of irregular operations in the hierarchy. In particular we have a series of base domains B^j as well as a series of versions of the original model $\mathcal{K}^j$ that are parameterized onto B^j and hence topologically equivalent.

The series of versions $\mathcal{K}^j$ are all irregular triangle meshes and are built under user control starting with the finest level $\mathcal{K}^J$ which is equal to the original model $\mathcal{K}$. Going from $\mathcal{K}^j$ to $\mathcal{K}^{j-1}$ could involve connecting two components or closing a detected tunnel. Typically smaller tunnels are closed first, but this is up to the user. The topology changes of the series of meshes $\mathcal{K}^j$ determine the topology changes of the later hybrid mesh $(\mathcal{M}^j, \mathcal{N}^j)$. In fact $\mathcal{N}^j$ will be topologically equivalent to $\mathcal{K}^j$. To allow for a level at which no irregular operations occur in the hybrid remesh, it is possible that $\mathcal{K}^{j-1} = \mathcal{K}^j$.

For example, for the Buddha model shown in the teaser figure, the original mesh $\mathcal{K} = \mathcal{K}^8 = \ldots = \mathcal{K}^4$ has genus six, $\mathcal{K}^3$ has genus three, $\mathcal{K}^2$ – genus two, and $\mathcal{K}^1 = \mathcal{K}^0$ is of genus zero, and thus can be parameterized onto a simple base mesh $\mathcal{B}^0 = \mathcal{N}^0$ consisting of 18 quads.

We next describe how the series of parametric domains B^j is created.

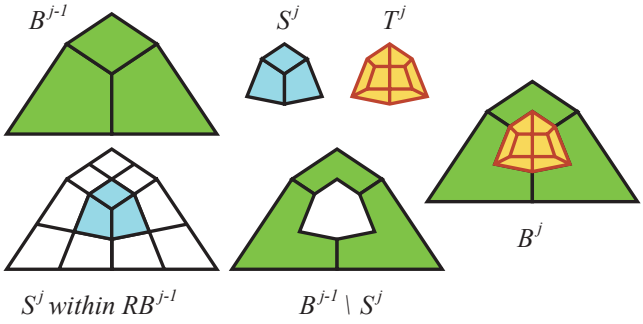


Figure 7: ***Repatching the parametric domain.***

Hybrid parametric domain. The series of parametric domains $\mathcal{B}^j$ is created recursively starting from $\mathcal{B}^0$. Suppose that on level j a submesh $\mathcal{S}^j$ is selected and replaced by a quad submesh $\mathcal{T}^j$, see Figure 7. Then we replace the portion of the parametric region corresponding to the set of quads in $\mathcal{S}^j$ by the quads in $\mathcal{T}^j$. At the same time, the parameterization of the surfaces outside of $\mathcal{S}^j$ does not change. Thus the parametric region for $\mathcal{K}^j$ is $\mathcal{B}^j = (\mathcal{B}^{j-1} \setminus \mathcal{S}^j) \cup \mathcal{T}^j$ for all $j = 1, \ldots, J$. Within the remesher framework this means that every time an irregular operation occurs, a new parameterization computation is required for the portion of the mesh that is repatched. At the same time, this reparameterization is local and does not affect the parameterization outside of the region affected by the irregular operation.

In our implementation, we delineate the area of the original mesh that corresponds to the repatched region parametrically, and create a separate copy of that piece of the original geometry which then goes into the patch layout generation stage.

Patch layout generation. Before the actual parameterization of the mesh regions can be performed we need to generate the patch layout, that is to split the corresponding mesh into a number of patches by a net of surface curves. This stage of the remeshing procedure is crucial for the quality of the resulting surface, and is therefore often performed with explicit user guidance in commercial remeshing tools [20] [19]. Creating a good patch layout is an acquired skill, and is often a part of the creative process in the design and special effects industries [24]. We use interactive patch layout specification both for the base mesh and for hybrid repatching operations (these typically happen near important geometric or topological features). This works well for the class of models we considered; however, for complex surfaces coming from medical imaging and scientific computing triangular hybrid meshes should be used together with an automatic patch layout generation procedure [8][25][15].

Our system provides a number of features that help the user in layout creation. For example, the system can replicate portions of the layout coming from the quad structure of the current level. This is useful for creating a buffer zone between the region of changed connectivity and surrounding unchanged areas. We also provide simple layout generating routines for cube growing as described in the following paragraph.

Cube building. If a patch has severe stretching, the system provides a semi-automated way for adding skin by building extra cubes. Consider a patch with a high stretch parameter and fix its boundary on the original mesh. The system will find the original mesh vertex in the patch which is furthest from the boundary in geodesic distance (Figure 8). It then traces four geodesic curves from the patch corners on the boundary to this extremal point. The right number of cubes to insert can be determined automatically by rounding the ratio of the average length of the traced curves to the average length of the four boundary curves. The vertices of the new cubes are then found by equally dividing the four curves traced to the extremal point.

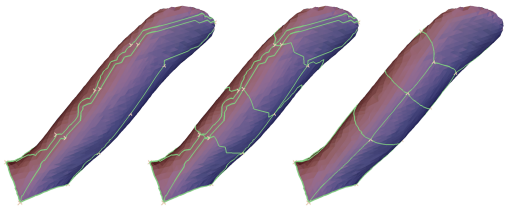


Figure 8: ***Building skin with cubes for the bunny ear. Left: geodesics to the furthest point, center: geodesics are split and connected, right: relaxed version.***

Buffer zone. In a regularly refined region of the hybrid mesh, the smoothness of the underlying parameterization is manifested through the uniformity of the surface sampling. A very non-uniform surface sampling near the boundary on finer levels of the hierarchy will arise if an irregular operation is performed in such a region. This can be somewhat alleviated by the introduction of a *buffer zone* of quads that are added to the area selected for repatching. This is similar to a boundary region in hierarchical spline representations [13]. The connectivity of the buffer zone comes from the quad arrangement of the current level of the hybrid hierarchy (before repatching), and does not change during repatching. However, the buffer zone does participate in layout relaxation steps performed for the selected region. In our implementation of the remesher the user can turn on and off the option to use the buffer zone for a particular irregular operation. The buffer zone is present only during remeshing to store the adjusted parameterization and is removed at the end of the remeshing process (this is possible because its connectivity matches the connectivity of the underlying regular level).

Parameterization and layout relaxation. The parameterizations within patches are computed using Floater's parameterization scheme [10] and a biconjugate gradient solver [12]. We relax the patch layout boundaries and corners similarly to the approach taken in [15]. The same relaxation procedure ensures a smooth parametric transition from repatched regions to surrounding areas when the patch layout in the buffer zone is relaxed.

Transferring layouts . Because the input mesh can be large, it is not efficient to run global relaxations on the original input mesh. Instead we run relaxations on a coarser version built through mesh simplification. To transfer a patch boundary back to the original model, we project all vertices onto the model, connect them with geodesics, and rerun the patch boundary relaxation. Global vertex relaxation is much more expensive and does not need to be rerun on the fine model. We typically build the hybrid mesh layout interactively on the coarse version and later transfer the entire layout off-line to the original model. If needed, the user can still make changes to the finest level layout. The final version of the Buddha in Figure 1 was generated using this method. The user worked with a mesh which had 10 times less vertices than the original. The interactive creation of hybrid layout structure on the coarser version took approximately two hours; the fully automatic transfer of the layout onto the finer model and construction of final hybrid remesh took another two hours.

3.2 Hybrid remeshing example

In this section we go in detail through the steps for building a hybrid mesh for a Max Planck head model, which does not require topological changes. This example illustrates basic steps of our hybrid remesher. We discuss topology changes in the next section.

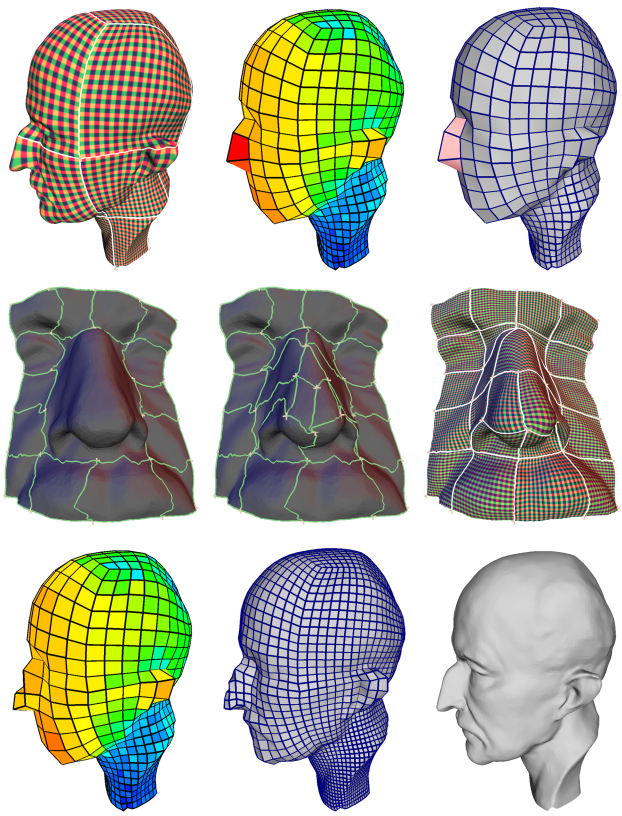


Figure 9: ***Sequence of screen shots for building the Max Planck hybrid mesh.***

The hybrid mesh is built in nine stages, see Figure 9.

- First we need a base domain $\mathcal{M}^0$. The user specifies four points on the boundary (bottom of the neck). Subsequently, the system automatically adds three cubes, places their vertices on the model, traces and smoothes the patch boundaries, and computes a parameterization for all patches as shown in the top left of Figure 9.
- Next the parameterization is used to compute three levels of regular refinement; this gives $\mathcal{M}^3$. For each quad q of $\mathcal{M}^3$, the system also computes a stretch parameter σ_q defined as $\sigma_q = A_q/P_q$ where A_q is the area on the input model covered by this quad patch and P_q is the area of the quad in parameter space. Assuming root quads to have unit parameter area, the parameter area P_q for any quad of level j is 2^{-2j}. The top middle shows a colored texture map indicating the value of σ_q for $\mathcal{M}^3$. In this way the user (or the system using a user-specified stretching threshold) can identify that the region around the nose needs extra skin.
- The user selects a region with 2×3 patches around the nose for an irregular operation (top right). The system automatically adds one layer of patches (giving 4×5 patches) as a buffer zone to ensure a parametrically smooth transition.
- The system cuts the selected 4×5 region from the original mesh, removes the patch layout in the 2×3 interior, and presents it to the user (middle left).
- The user draws a new patch layout in the empty region by specifying control points and their connectivity. The system connects the control points with geodesics.
- After the layout is specified, the system smoothes the curves and computes a parameterization for the new patches (middle right).
- The system attaches the newly created patches at level 3 of the remesh replacing the quads selected earlier, thereby going from $\mathcal{M}^3$ to $\mathcal{N}^3$. The patches outside the selected area get refined automatically. The stretch parameters are recomputed for the quads of $\mathcal{N}^3$ (bottom left). Compare with the top middle figure to see how stretching around the nose has been reduced.
- Remeshing now continues to build $\mathcal{M}^4$ (bottom middle). Around the nose, the new parameterization is used, everywhere else we still use the original one.
- More adaptive refinement steps are added until we reach the final adaptive remesh $\mathcal{M}^8$. The remesh has $26K$ vertices compared to $25K$ for the original. The relative L_2 error is 0.01%.

Note that for this example the user only needed to specify a patch layout in the small area around the nose, while the rest of the steps were taken automatically by the system. The stretched region selection can be done automatically by the system using a threshold, however we found that experienced users of our system often prefer to perform selection manually as it gives fuller control, especially for a simple model such as the human head.

3.3 Dealing with changes in topology

In this section we discuss how hybrid remeshing accommodates topological changes.

The first task before creating a hybrid mesh for a model with non-zero genus is to find all the handles/tunnels and decide were to perform the non-separating cuts and cap insertions that would reduce the genus. The system finds all the non-obvious tunnels in the original mesh using the approach of [16]. That procedure gives an approximate location of handles. The user has an opportunity

to specify the precise loop along which a cut will be performed and specifies on which level a particular handle becomes active. The system then removes all the specified handles to obtain the $\mathcal{K}^0$ mesh for which the base mesh is built. Remeshing proceeds from one level to the next building parameterizations and creating levels of hybrid hierarchy. One feature of hybrid mesh creation with topological changes is that when we come to a level where a particular handle becomes active, the irregular operation around the cut needs to be performed. The system selects the quads that cover the cut on the current level, and the corresponding mesh region is extracted from the original mesh (that region will have the handle cut with caps covering the sides of the cut). The handle is activated in the extracted mesh region: the caps are removed, and the sides of the cut are stitched together by the system. The modified mesh region is presented to the user for layout creation.

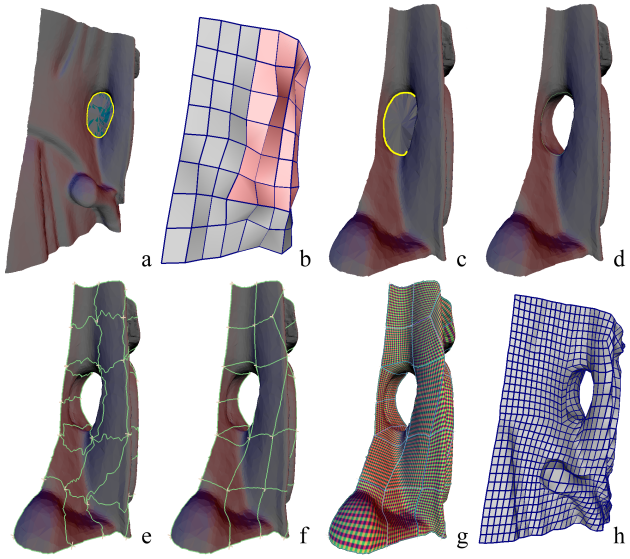
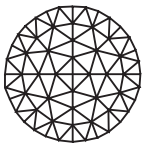


Figure 10: ***Sequence of screen shots for building the side of the Buddha.***



Hole stitching. In order to close a tunnel in the original mesh, the user delineates a closed loop (non-separating cut) that goes around a handle and the system cuts the original mesh along that loop and then uses the following hole stitching procedure to close the resulting holes with caps. The input to the hole stitching consists of a closed polyline in 3D space: first, the average position of the vertices in the given loop is computed, and then a number of "concentric" loops are built in a recursive fashion: we obtain the next loop by removing every other vertex in the previous loop and then placing the vertices on the segment between the vertex position on the previous loop and the center. So after a logarithmic number of steps we come to a small loop surrounding the center vertex. As the last step all the vertices in that loop are connected to the center. The resulting mesh has connectivity similar to the one shown on the left, and has a property that all the vertices within the generated region have small valence. If needed a relaxation procedure can be applied to smooth out the cap region.

Example: Close-up of region on the Buddha. In this section we show how a tunnel on the side of the Buddha was handled, see Figure 10.

- The user draws a curve along the tunnel and the system closes the tunnel with two filled caps (Figure 10a).
- Once all tunnels are closed or handles are cut, a parameterization onto a genus 0 base domain can be computed and remeshing can begin. All filled caps get parameterized as well. Note that the two back to back filled caps that were put in for each cut get assigned to different regions in the parametric domain. At the appropriate level, a genus change is introduced. The system selects the appropriate region on the remesh (Figure 10b), which covers both filled caps. Note that in general there could be two separate regions, one for each of the caps.
- The system cuts out the submesh of the original surface which is associated with this region (Figure 10c) and removes the caps (Figure 10d).
- An entirely new patch layout needs to be generated for this submesh, with the only requirement that it matches the boundary of the cut-out region (Figure 10e). The new layout is either drawn by the user from scratch, or can be automatically copied from a layout template (see "Transferring Layouts" in Section 3.1).
- The specified layout is automatically relaxed (Figure 10f), the corresponding submesh parameterized (Figure 10g), and the highlighted region of Figure 10b replaced by the set of newly created root quads.
- The remeshing process continues (Figure 10h). The remesher uses the original base domain parameterization outside of the pink region, and the new parameterization for the submesh.

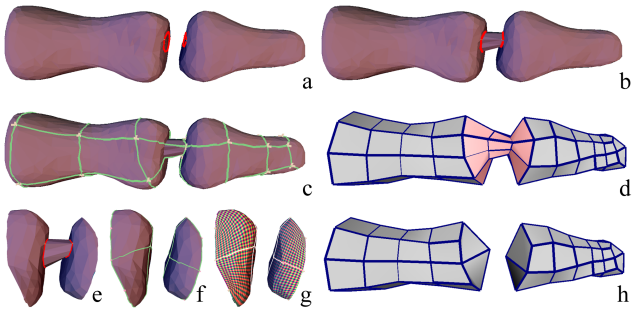


Figure 11: ***Remeshing of multiple components.***

Multiple components. Models with multiple components are handled in a manner similar to non-trivial genus models. The main difference in this case is that instead of cutting the tunnels we need to connect neighboring components. This is done with the following procedure, illustrated in Figure 11.

The user starts by outlining a simple closed curve on each original component (Figure 11a). The system creates a cylinder in between (Figure 11b). Then a patch layout is generated for the combined components (Figure 11c). After one level of refinement, the user decides to make a cut and highlights the region around the tunnel (Figure 11d). The system cuts out the corresponding region of the original mesh (Figure 11e) and removes the cylinder. The user provides new patch layouts for each of the caps (Figure 11f) and the system computes parameterizations (Figure 11g). The cut can now be made and remeshing can continue (Figure 11h).

3.4 Remeshing results

We used our remeshing system to build hybrid meshes for a variety of irregular meshes. In this section we present several models and discuss hybrid mesh hierarchies built for them, as well as provide some statistics of the remeshing process. We conclude with

some compression results.

Feline. The results for the Feline model are shown in Figure 12. We use the cube building algorithm to construct a 4 cube base domain, insert legs and tail on level 1, and horns on level 2. The original tail has two tunnels which get recreated on levels 2 and 3, see Figure 13.

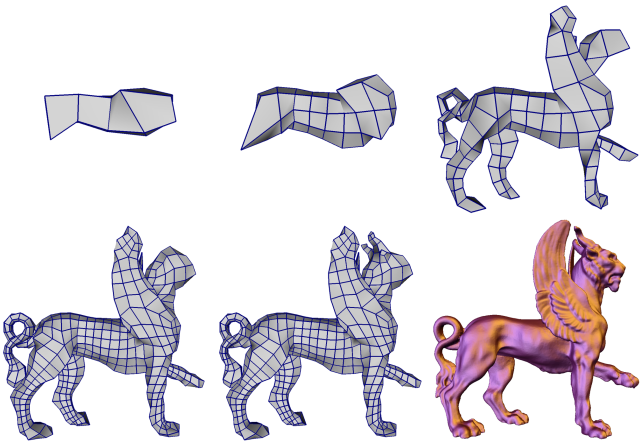


Figure 12: ***The Feline remesh begins with 4 cubes adding long chains of cubes automatically at level 1 to accommodate legs, tail, head and wings. At level 2 horns are added and the tail fuses (see Figure 13). The bottom right shows the adaptively refined $\mathcal{M}^8$ with a relative L_2 error of 0.01%.***

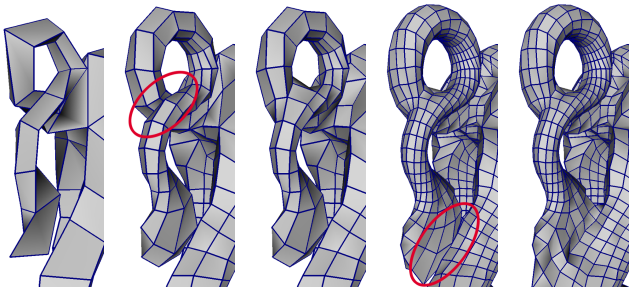


Figure 13: ***Detail of the Feline tail section. On level 1 a long cube chain is grown to accommodate the tail. At level 2 the top loop of the tail fuses, while at level 3 the end of the tail fuses with the leg.***

David head. The David head starts with a 2 cube base domain (Figure 16, left). Irregular operations are used to add skin for the nose and align patch boundaries with the eye lids, see Figure 14. Figure 15 shows a close-up around the eye while Figure 16 shows the uniform $\mathcal{M}^6$ as well as the final adaptive $\mathcal{M}^{11}$.

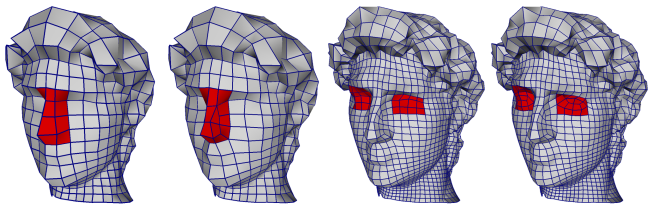


Figure 14: ***Irregular operations for David's nose at regular refinement level 3 (left; before and after) and for his eyes at level 4 (right; before and after).***

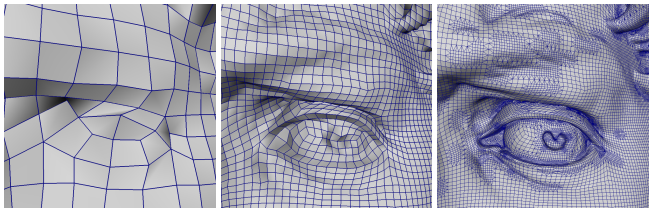


Figure 15: ***Detail of David's left eye at remeshing levels 4 (after insertion of hybrid patches for lid alignment), 6, and 11 (adaptive).***

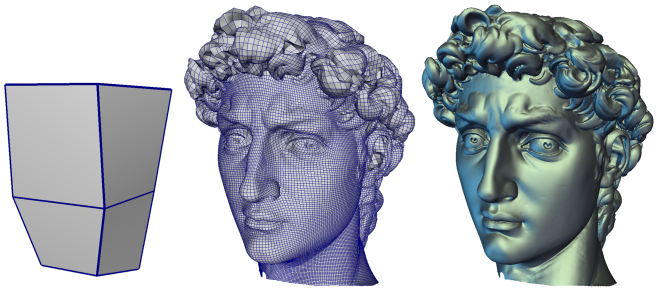


Figure 16: ***Hybrid mesh of David's head. Coarsest level (two cubes with open bottom), level 6 after hybrid operations (see Figure 14, and the final remesh with 11 levels (see eye detail in Figure 15, right).***

Foot bones. The foot bones are an example with 19 connected components, see Figure 17. We already discussed the procedure in detail for two bones, see Section 3.3 and Figure 11. For the entire model, Figure 17, the user first adds cylinders to group the components into a single component. The base domain is laid out by hand (top left), except for the digits which are done by cube building. During various stages of the remeshing, cuts are made to create new components as described above.

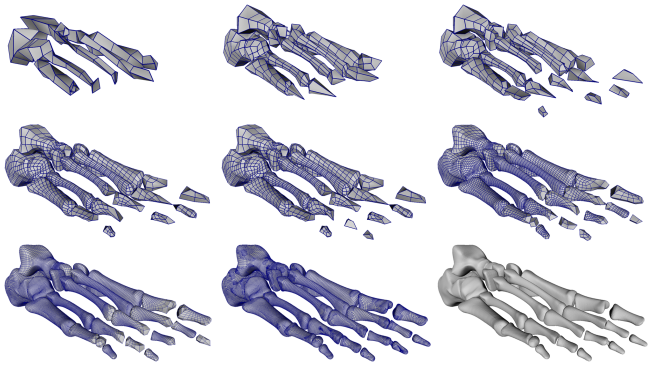


Figure 17: ***Foot bones example: remeshing of a multiple component dataset starting from a single connected component base mesh.***

Statistics. Table 1 shows various statistics for the models used in this paper. For big meshes (Buddha and David head) the computation time is the time needed to transfer the patch layout to the original model together with the time needed to compute the fine remesh. The error is computed with the I.E.I-CNR Metro tool (the Metro tool was unable to compute the error for the finest level Buddha).

Note that the original David data is contaminated with "topological noise," i.e., there are numerous tiny tunnels, especially around the hair. These tunnels were automatically removed with a topological noise removal algorithm described in [16]. After this topolog-

ical cleanup step there are still significant "stalactite" like features left which are mostly located *inside* the model. These stalactites were removed during hybrid remeshing.

Model	# Vertices Original	# Vertices Remesh	Relative L_2 error	Computation Time
Buddha	544171	306644	N/A	2 hr*
Planck	25445	26067	1.0e-4	2 min
Feline	49864	70793	9.2e-5	14 min
David	590337	380582	4.0e-4	43 min
Bones	33684	75747	7.3e-5	12 min

Table 1: ***Summary of hybrid mesh results for different models. The relative L_2 errors are computed with the I.E.I.-CNR Metro tool. Timings were performed on a 1Ghz Pentium III machine. *The Buddha timing was performed on a 2Ghz Pentium 4 machine.***

4. HYBRID MESH PROCESSING

The multiresolution structure of hybrid meshes allows for a conceptually easy extension of multiresolution processing tools from the semi-regular setting: the bulk of processing in regularly refined areas stays the same, while a proper analogon of the operation is needed in irregular portions of the mesh. We give more details on how to extend wavelet based compression tools to hybrid surface representations.

4.1 Multiresolution: Subdivision and Wavelets

Because our hybrid meshes consist of a preponderance of regular operations, they naturally support subdivision. Namely, subdivision can be performed during a regular refinement step $\mathcal{N}^{j-1} \rightarrow \mathcal{M}^j$ in exactly the same way as in the standard semi-regular case. If it is followed by any irregular step $\mathcal{M}^j \Rightarrow \mathcal{N}^j$, the next regular refinement $\mathcal{N}^j \rightarrow \mathcal{M}^{j+1}$ applies subdivision rules as if $\mathcal{N}^j$ were a base mesh, and so forth.

Denote by $S(B)$ the space of limit surfaces with the same connectivity as the base mesh B and $S^h(B)$ be the space of hybrid limit surfaces. Since semi-regular subdivision is a hybrid subdivision with no irregular refinement we have $S(\mathcal{N}^0) \subset S^h(\mathcal{N}^0)$. Let J be the finest level where irregular operations were performed. Obviously, we have $S^h(\mathcal{N}^0) \subset S(\mathcal{N}^J)$. This inclusion implies that all standard smoothness analysis can be automatically applied to hybrid subdivision with a fixed number of irregular operations. For local subdivision schemes this condition can be relaxed. Namely, a limit surface is smooth at a point p if some neighborhood of p has only a finite number of irregular operations. The meaning of this restriction is that an irregular step can alter the surface in an arbitrary way but after some number of pure (locally) regular refinements the smoothness will be restored. One should keep in mind, that irregular operations may introduce new irregular vertices, which are also irregular in $S(\mathcal{N}^J)$.

Similar to subdivision, we can extend the construction of wavelets. To perform one analysis step $\mathcal{N}^j \rightarrow \mathcal{N}^{j-1}$ we first undo any irregular operations and store the difference between $\mathcal{N}^j$ and $\mathcal{M}^j$. Then we apply any chosen semi-regular wavelet construction to decompose $\mathcal{M}_j$ into the coarser mesh $\mathcal{N}^{j-1}$ and a collection of wavelet details W^{j-1}. In the next section we describe a particular wavelet approach we use in our geometric compression tool.

4.2 Compression of Hybrid Representations

We use a compression approach similar to that of Khodakovsky et al. [21]. The main components of that algorithm are a wavelet transform and a zero-tree coder. We use wavelets designed using a "quadrature mirror filter" (QMF) construction based on Catmull-Clark subdivision. This is a well known technique for designing wavelets in 1D: A high pass filter is constructed as a reflection of a lower pass filter shifted by one node and with flipped signs at all odd locations. Since the Catmull-Clark scheme is a tensor product of two cubic $1D$ splines inside a regular patch, QMF filters are immediately generalized to the surface setting. This defines semi-orthogonal (in coefficient norm) compactly supported wavelets. We extend wavelets around extraordinary vertices with the same technique as in [21]. Dual wavelets are not compactly supported so an encoding step involves solving a sparse linear system. The system is well conditioned. In all our experiments it took less then one minute to solve. Finally, wavelet coefficients are encoded with a zero-tree coder (similar to [21]). As an alternative the wavelet construction of Bertram et al. [6] could also be used.

The other necessary component of the hybrid compression algorithm is encoding of a transition $N^j \Rightarrow M^j$, that is, storing the repatching information. It involves encoding submesh connectivity and geometry in a way similar to compressing the coarsest level of a mesh hierarchy [32] [4]. Also, faces to be removed from the main mesh during reconstruction must be encoded. The latter is done by encoding a seed edge and then encoding the boundary of a region to remove. Along with the seed edge we store the corresponding submesh edge to provide the decoder with the correspondence information. This process is done for all connected components of the boundary. The geometry of the repatched region is stored in a relative frame with respect to its boundary.

During reconstruction we have the freedom of not applying any particular irregular operation but continue with pure regular refinement. (For consistency we should skip all irregular operations at finer levels too.) This operation is very useful during progressive reconstruction. At very low bitrates we may want to present only a very approximate shape of the model with greatly simplified topology. At progressively higher rates more and more features are introduced through "turning on" the irregular steps, see Figure 18.

4.3 Results

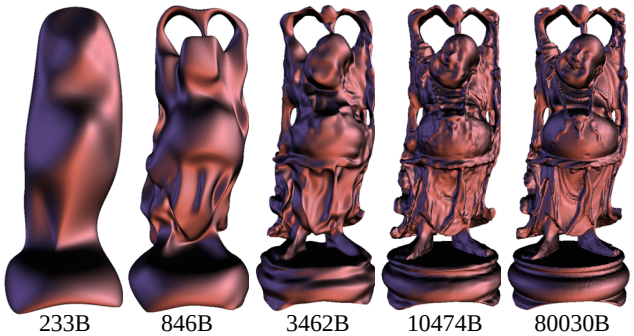
Figure 18 shows partial reconstructions of the Buddha model using our progressive compression algorithm. The reconstruction process starts from a semiregular mesh. Since at very low bitrates the shape of the model is very approximate there is no need for any information about irregular operations. The encoder will compresses repatching information about a particular scale after it encodes some information about a geometry at this scale. Figure 18 illustrates this process. Irregular operation are introduced on the second and third reconstructions.

The numbers under the pictures show sizes of compressed files in bytes and reconstruction errors measured with the Metro tool. Since we had a problem with running the Metro tool on the finest original model we used a simplified version as a reference model for error computations.

5. CONCLUSIONS AND FUTURE WORK

We presented a novel approach for representing complex geometric shapes that extends semi-regular mesh representations to include topological and parametric changes within the mesh hierarchy. A user guided remeshing procedure was introduced that can produce this novel hybrid mesh representation from scanned surface data. The processing tools for the proposed hybrid meshes can be extended from their semi-regular counterparts, and applied for editing, rendering, and compression of hybrid representations. This paper presents some initial compression results.

In this work, we did not show any high topological complexity



233B e=100 | 846B e=28 | 3462B e=12 | 10474B e=3.6 | 80030B e=2.4

Figure 18: ***Partial reconstructions of the compressed Buddha model. Numbers under the pictures show the size of compressed files and reconstruction error in units of*** 10^{-4} ***of the bounding box diagonal. Note, how topological changes appear during progressive reconstruction.***

models such as isosurfaces from medical imaging or scientific computing. The extension of the presented approach into a system that can handle such complex models fully automatically is an exciting project for future work. Another related problem is the approximation of dynamically changing, topologically complex, geometry. In both of these cases, triangular hybrid meshes will be preferable as they would provide better mesh quality for complex surfaces.

Acknowledgements. This work was supported in part by DARPA and NSF (DMS-9875042) and NSF (CCR-0133554, ACI-9982273, DMS-9872890). Additional support was provided by the Packard Foundation, Alias|Wavefront, Microsoft, nVidia, Pixar and Intel. Models are courtesy Cyberware, Max Planck Institute for Computer Graphics and Stanford University.

6. REFERENCES

[1] E. Akleman and J. Chen. Guaranteeing 2-manifold property for meshes. In *Proceedings of the Shape Modeling International*, pages 18–25, 1999.

[2] E. Akleman, J. Chen, and V. Srinivasan. A new paradigm for changing topology during subdivision modeling. In *Proceedings of Pacific Graphics*, 2000.

[3] P. Alliez and M. Desbrun. Progressive compression for lossless transmission of triangle meshes. *Proceedings of SIGGRAPH*, pages 195–202, 2001.

[4] P. Alliez and M. Desbrun. Valence-driven connectivity encoding of 3d meshes. In *Eurographics 2001 Conference Proceedings*, pages 480–489, Sept. 2001.

[5] C. Barghiel, R. Bartels, and D. Forsey. Pasting spline surfaces. In *L. Schumaker, M.Daehlen, and T.Lyche, editors, Mathematical Methods for Curves and Surfaces, pages 31–40.* Vanderbilt University Press, 1995.

[6] M. Bertram, M. A. Duchaineau, B. Hamann, and K. I. Joy. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization. In *Proceedings of IEEE Visualization*, pages 389–396, 2000.

[7] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of SIGGRAPH*, pages 317–324, 1999.

[8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH*, pages 173–182, 1995.

[9] J. El-Sana and A. Varshney. Topology simplification for polygonal virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):133–144, 1998.

[10] M. S. Floater. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231–250, 1997.

[11] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH*, pages 209–216, 1997.

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[13] C. Gonzalez-Ochoa and J. Peters. Localized-hierarchy surface splines (less). *ACM Symposium on Interactive 3D Graphics*, pages 7–16, 1999.

[14] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. *Proceedings of SIGGRAPH*, pages 325–334, 1999.

[15] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder. Normal meshes. *Proceedings of SIGGRAPH*, pages 95–102, 2000.

[16] I. Guskov and Z. Wood. Topological noise removal. In *Proceedings of Graphics Interface*, pages 19–26, 2001.

[17] H. Hoppe. Progressive meshes. *Proceedings of SIGGRAPH*, pages 99–108, 1996.

[18] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of SIGGRAPH*, pages 295–302, 1994.

[19] http://www.headus.com.au/. Headus cyslice.

[20] http://www.paraform.com/. Paraform.

[21] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. *Proceedings of SIGGRAPH*, pages 271–278, 2000.

[22] L. Kobbelt. $\sqrt{3}$ subdivision. *Proceedings of SIGGRAPH*, pages 103–112, 2000.

[23] L. P. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3):249–260, 2000.

[24] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. *Proceedings of SIGGRAPH*, pages 313–324, 1996.

[25] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH*, pages 95–104, 1998.

[26] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. *Proceedings of SIGGRAPH*, pages 131–144, 2000.

[27] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH)*, 21(4):163–169, 1987.

[28] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, 1997.

[29] J. Peng, V. Strela, and D. Zorin. A simple algorithm for surface denoising. In *Proceedings of IEEE Visualization*, pages 107–112, 2001.

[30] J. Popovic and H. Hoppe. Progressive simplicial complexes. *Proceedings of SIGGRAPH*, pages 217–224, 1997.

[31] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[32] C. Touma and C. Gotsman. Triangle mesh compression. *Graphics Interface 98 Conference Proceedings*, pages 26–34, june 1998.

[33] L. Velho and D. Zorin. 4-8 subdivision. *Computer Aided Geometric Design*, 18(5):397–427, 2001.

[34] D. Zorin and P. Schröder, editors. *Subdivision for Modeling and Animation*. Course Notes. ACM Siggraph, 2000.