Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

Progressive Multiresolution Meshes for Deforming Surfaces

Scott Kircher† and Michael Garland‡

University of Illinois at Urbana–Champaign

Abstract

Time-varying surfaces are ubiquitous in movies, games, and scientific applications. For reasons of efficiency and simplicity of formulation, these surfaces are often generated and represented as dense polygonal meshes with static connectivity. As a result, such deforming meshes often have a tremendous surplus of detail, with many more vertices and polygons than necessary for any given frame. An extensive amount of work has addressed the issue of simplifying a static mesh; however, these methods are inadequate for time-varying surfaces when there is a high degree of non-rigid deformation. We thus propose a new multiresolution representation for deforming surfaces that, together with our dynamic improvement scheme, provides high quality surface approximations at any level-of-detail, for all frames of an animation. Our algorithm also gives rise to a new progressive representation for time-varying multiresolution hierarchies, consisting of a base hierarchy for the initial frame and a sequence of update operations for subsequent frames. We demonstrate that this provides a very effective means of extracting static or view-dependent approximations for a deforming mesh over all frames of an animation.

1. Introduction

Complex time-varying surfaces arise in countless applications. By their very nature, the center of attention in any movie or game is invariably in motion. Isosurface animation is a core method for visualizing dynamic scientific simulations. Whether produced by hand, as in key-framing, via simulation of physical processes, as with cloth and skin, or acquired via motion capture, time-varying surfaces are very often represented by polygon meshes.

In many cases, it is advantageous to use a single static mesh connectivity when modeling surface deformations rather than using a separate mesh for each time step. It requires significantly less storage space and generally makes processing the time sequence much more tractable. Many time-stepping finite element simulations also have a strong preference for maintaining a fixed mesh connectivity, as altering the connectivity can involve an expensive reprojection of the solution to the new mesh. Yet using a fixed connectivity also has an obvious drawback: it can require far more polygons than would be necessary in any given frame. This is particularly true for surfaces that undergo extremely non-rigid deformations, such as might occur in cloth simulations, morphing, and other non-skelatal forms of animation. Such deformations require extremely dense meshes, as the static connectivity must be able to accurately represent all possible deformations of the surface.

It is thus very common that meshes used in generating these deformations have entirely too many triangles in any particular frame. For static surfaces, there is a wide selection of simplification methods that can be used to remove unnecessary mesh detail. This is emphatically *not* the case for time-varying surfaces. Very little work has addressed how to maintain accurate approximations of a time-varying surface, let alone a multiresolution hierarchy that can be used for efficient adaptive refinement.

Producing a single coarse connectivity can lead to arbitrarily bad approximations in certain frames when the surface deformation is highly non-rigid (see Figure 1). A single multiresolution hierarchy yields similarly bad results—very deep traversals to meet a given error tolerance or very bad approximations for a given triangle budget. On the other hand, producing an entirely new approximation for each frame wastes a great deal of space and, having no tempo-

† e-mail: kircher@uiuc.edu
‡ e-mail: garland@uiuc.edu

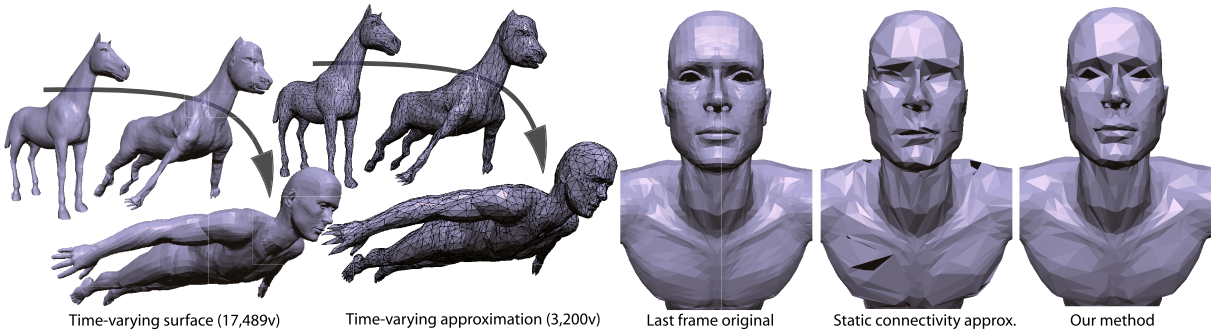


Figure 1: *Approximating a horse-to-man animation. A single approximation is not suitable for all frames of this highly non-rigid deformation. Our method dynamically adjusts an entire multiresolution representation over time, producing high quality approximations at any level-of-detail, on every frame.*

ral coherence, can cause the surface to twitch and vibrate in a most unpleasant manner.

We therefore propose a new multiresolution representation for deforming surfaces that can provide high quality surface approximations at all frames of an animation. We define a new mesh structure—the *multilevel mesh*—that represents the input surface at multiple levels of detail. We have developed a dynamic reclustering scheme that can incrementally update this hierarchy in response to the geometric deformation of the underlying surface. By recording a base hierarchy for the initial frame, and a sequence of incremental update operations, we arrive at a new progressive representation for time-varying multiresolution hierarchies. We demonstrate that this provides an efficient means of extracting high quality static or view-dependent approximations for a deforming mesh over all frames of an animation.

2. Related Work

Simplification and LOD. There is now extensive literature on the approximation of dense polygonal meshes by coarser meshes that preserve surface detail [Gar99, LRC*02]. These methods may be used in rigid motion animations. However, as they coarsen a mesh based on a specific fixed shape, the meshes they produce can yield very poor approximations if the surface deforms non-rigidly.

The simplification methods in widespread use are predominantly iterative edge contraction methods; representative algorithms include those due to Hoppe [Hop96] and Garland & Heckbert [GH97]. Such methods induce a simple multiresolution structure on the surface that can be used for adaptive refinement of the mesh [XV96, Hop97, LE97]. With a single hierarchichal structure, an application can extract many possible approximations of the input surface. This has been used most frequently for performing view-dependent refinement for real-time display. But as with the simplification methods upon which they are based, this approach uses a single static hierarchy whose suitability degrades quickly if the surface deforms.

Time-Varying Approximation. Compared to the simplification of static meshes, the problem of producing good approximations for time-varying meshes has gone largely unaddressed. One natural approach is to augment standard simplification algorithms to consider all frames of an animation when choosing an edge to contract. This is the approach taken by Mohr and Gleicher [MG03], who adapt the QSlim algorithm [GH97] by summing quadrics over each frame of the animation. The result is a single mesh that attempts to provide a good "average" approximation over all frames. This approach can produce acceptable results, provided that the surface does not deform too much, in which case the "average" best fit mesh tends to be uniformly poor in all frames. DeCoro and Rusinkiewicz [DR05] put forth a related method specific to linear-blend skinned models. This method works quite well, but is limited to a very specific class of deformations.

The only scheme that we are aware of that provides a multiresolution format for deforming surfaces is the Time-dependent Directed Acyclic Graph, or T-DAG, construction introduced by Shamir *et al.* [SPB00, SP01]. The T-DAG is built by merging the individual multiresolution hierarchies for each frame of a mesh sequence together into a unified graph. Tags are assigned to each node specifying the time interval over which the node should be alive. The T-DAG has the advantage of being able to handle arbitrary topology changes as well as geometric deformations. The primary drawback of this approach is that it is inherently non-incremental and potentially space-inefficient. Although the T-DAG can be constructed incrementally by taking history into account when performing the mesh decimation for the next frame, this can worsen the subsequent approximations. Another drawback is that it appears to be difficult to maintain an arbitrary cut through the alive portion of the T-DAG as the animation proceeds, necessitating an inefficient re-traversal

from the roots [SPB00] or the use of the same vertex hierarchy for all time-steps [SP01].

A problem related to approximating deforming meshes arises in mesh morphing. If the source and target meshes do not have identical connectivity, the source cannot be transformed into the target simply by interpolating vertex positions. One standard solution to this problem is to compute a single static *metamesh* [LDSS99] representing the union of both meshes. This typically produces a mesh of much higher complexity than the inputs. A more economical solution is to compute a sequence of vertex additions, vertex removals, and edge flips that will transform the source into the target [LL05, AL02, ALS04]. This incremental updating of the mesh connectivity parallels our own work. But whereas these morphing methods attempt to derive a good sequence of operations to achieve a specified connectivity, our goal is to adapt the connectivity to produce a good surface approximation.

Clustering. Clustering is an attractive algorithmic paradigm that is widely used in countless computational disciplines. Of particular interest to us here is that iterative edge contraction and their resulting hierarchies can be viewed as clustering operations [Gar99]. Individual contraction operations merge vertex clusters into larger clusters. The well-known vertex hierarchies used for view-dependent refinement simply encode the hierarchical nesting of these clusters.

The key to our work is that we view the problem of maintaining a good multiresolution hierarchy as the surface deforms as one of reclustering. A hierarchy built for the first frame represents a hierarchical clustering of the initial mesh. As the surface deforms, we will seek to incrementally improve this clustering. Our approach to multilevel reclustering is inspired by the well-known Kernighan-Lin partition refinement algorithm [KL70]. In particular, we draw on the hierarchical variants of the Kernighan-Lin algorithm that have been developed for rapid graph partitioning in the distributed computing field [KK98a, KK98b].

Carr and Hart [CH04] introduced a method of re-clustering a vertex hierarchy utilizing tree rotations and grandchild swapping. However, to be valid, the clusters must remain connected. This property can be violated by the tree update operations. Carr and Hart reduce the probability of invalid tree updates with the use of Bloom filters. However, their probabilistic method—which they developed for texture atlas rebalancing—is not suitable for mesh approximation, as it cannot guarantee correctness. Indeed, by defining a rather different framework for representing and updating the hierarchy, we are able to guarantee that only valid updates are performed.

3. The Multilevel Mesh

Our *multilevel mesh* is constructed by iterative edge-contraction on an initial mesh M_0. If M_1 is the simplified

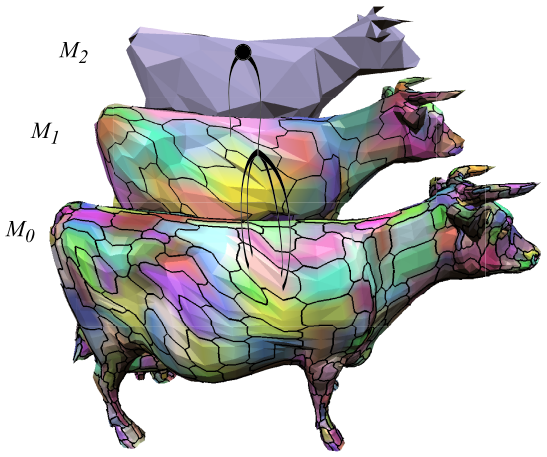


Figure 2: *The multilevel mesh hierarchy. Mesh edges encode intralevel connectivity, while contraction edges (thick arcs between meshes) record dependencies between levels. Groups of vertices that all have the same parent form a contraction cluster (depicted by their boundaries).*

output, we define a set of *contraction edges* connecting each vertex a in M_1 to all the vertices in M_0 that were contracted together to form a. By repeating this process we can construct additional levels, M_2, M_3, ..., M_n, and the contraction edges connecting each level to the one above it. In particular, we use the QSlim algorithm for this process [GH97]; and we choose the vertex count of each successive level so as to have a constant complexity reduction factor at each level. Note that this factor, the *branching factor*, need not be equal to two.

The multilevel mesh structure consists of a number of *levels*, connected by contraction edges. Each level is, itself, a mesh, complete with *vertices* and *mesh edges*. Thus, there are two distinct kinds of edges in the multilevel mesh. The contraction edges indicate which finer vertices were contracted together to form each coarser vertex, whereas the mesh edges indicate the vertex connectivity within a single level (See Figure 2). We do not require any level (including the original input) to be manifold. Levels are numbered in increasing order from fine to coarse, starting with M_0.

If there is a contraction edge connecting vertex v in level M_k to a vertex a in M_{k+1}, then a is called the *parent* of v. Likewise, v is a *child* of a. The set of all children of a particular vertex, a, is a *contraction cluster*, denoted by C_a. Note that a corresponds to a connected patch (namely C_a) on M_k, and also on all lower levels.

Given only the finest level mesh M_0, and the contraction clusters at each level, we can reconstruct the vertex positions and connectivity of each higher level (this will become important when dynamically modifying the hierar-

chy). In particular, since we are using the Quadric Error Metric (QEM) [GH97], each vertex, v, in M_0 will have an associated quadric, Q_v, computed directly from the input mesh. Each vertex, a, in level M_k for $k > 0$ will have an associated quadric Q_a computed by $Q_a = \sum_{u \in C_a} Q_u$. The optimal position of vertex a, is then obtained by minimizing $Q_a(a)$. The *error* of vertex a is denoted $E_a = Q_a(a)$.

The connectivity of M_k is formed directly from the connectivity of M_{k-1} and the contraction edges connecting the two levels. Given a polygon p in M_{k-1}, we map each of its vertices to its parent in M_k. If p does not degenerate through this mapping (i.e. it still has three or more distinct edges), then it exists in M_k.

The basic structure of our hierarchy is similar to that of vertex hierarchies [XV96, Hop97, LE97], in that it has nodes that represent some kind of contraction operation, with arcs that represent the dependencies between operations. However, our multilevel mesh additionally has mesh connectivity at each level of the hierarchy, and we view the nodes as vertices at a particular resolution, rather than simply contraction operations. If the mesh edges are discarded, the structure is completely compatible with existing view-dependent-refinement methods. However, this mesh connectivity is vital in our reclustering algorithm, as it provides the means to determine the validity of an update (§4.2).

4. Reclustering

A multilevel mesh is constructed from a specific input mesh. However, for a deforming surface this is inadequate, as discussed in §1. In particular, since the coarser vertex positions are based on the quadrics of their children, these positions can become arbitrarily bad as the children move. In addition, the simplified connectivity may become ill-suited for the new shape. Thus, as the underlying surface deforms, we seek to improve the hierarchy to better represent it. We do so by modifying the contraction clusters to minimize the total error of the hierachy. To simplify the discussion, we will first describe this improvement process from the standpoint of a two-level multilevel mesh $\{M_0, M_1\}$ representing a static object. Although the "metric" referred to in subsequent sections can be fairly arbitrary, in practice we use the QEM [GH97] (and a hierarchical version of it, to be introduced in §4.3), because it provides a good trade-off between speed and accuracy, and has a nice additive property that makes updating easier.

4.1. The Swap Operation

The fundamental reclustering operation we have chosen is the *swap* operation, defined as follows. Consider a vertex, $v \in M_0$, in some contraction cluster C_a, which has a neighbor (defined by a mesh edge) in contraction cluster C_b. Except in certain situations (§4.2), v can be moved to cluster C_b (Figure 3). This operation is a swap, identified by the triplet

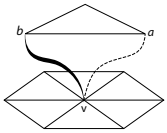


Figure 3: *Swap (v,a,b) changes v's parent from a to b.*

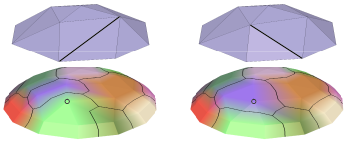


Figure 4: *A swap can cause an edge flip in the approximation. The adjacency of clusters at level k governs the mesh connectivity of level $k+1$.*

(v,a,b). When a vertex moves from one cluster to another, this is nothing more than changing its parent. This change, however, will affect the value of the error metric (and therefore the optimal vertex positions). It may also affect the connectivity of M_1 (Figure 4). In particular, note that the swap can produce an edge flip in the approximation (something not possible in solely adaptive refinement of a static hierarchy).

There are a number of ways that clusters could be improved; however, we have chosen the swap operation because it is the fundamental partition refinement operation used in Kernighan-Lin based algorithms, which have been very successful in other domains. The swap operation is suitable as a fundamental reclustering operation for many of the same reasons that edge-collapse is suitable as a fundamental simplification operation. A swap is a very fine-grained operation with a well-defined effect on the clustering (and hence the hierarchy that we are improving). In addition, other kinds of operations, like moving whole groups of vertices or merging and splitting clusters, can be performed as a sequence of swaps. More free-form approaches, like Lloyd-style relaxation [SWG*03, CH04], would be difficult to do efficiently in a hierarchical setting.

4.2. Swap Priority and Validity

Only certain swaps are both *valid* and *beneficial*. Our reclustering algorithm processes only valid swaps in the order of greatest to least benefit.

We consider a swap to be invalid if it would cause any cluster to become disconnected. By disallowing invalid swaps, we guarantee that every cluster is a connected set of vertices, thus ensuring that the induced approximation could have been arrived at by a sequence of edge contractions on the original mesh. This validity rule holds regardless of whether the surface is manifold.

A potential swap (v,a,b) is valid if v is not the only child of a and it is not a *pinch vertex*. A vertex v is a pinch vertex if the vertices that share a face with v and are in the same cluster cannot be formed into a single chain (or tree, if the region is non-manifold) of edges, *without* passing through v. If v is a pinch vertex, swapping it from C_a to C_b will disconnect the cluster C_a; Figure 5 illustrates this case.

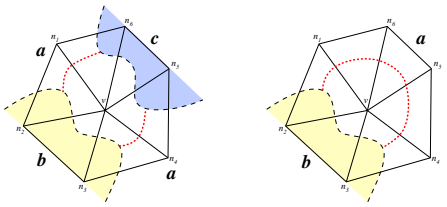


Figure 5: *Pinch (left) and non-pinch (right) vertices. Dotted red lines indicate boundary of cluster C_a after swap (v,a,b). In the pinch vertex case, the boundary becomes disconnected.*

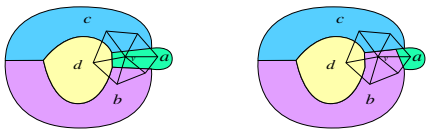


Figure 6: *Topology preservation. Performing swap (v,a,b) connects C_b and C_c, but since they are already adjacent, the approximation becomes nonmanifold.*

The validity rule just described guarantees that all clusters remain connected, regardless of surface topology. However, it does not guarantee that approximations before and after swapping are homeomorphic (e.g., initially manifold regions may become non-manifold). For some applications, notably texture mapping, maintaining a homeomorphic correspondence between manifold approximations is crucial. Thus, we provide a second optional validity rule that can be applied if the user requires topological preservation.

We assume that M_0 is manifold and that a homeomorphism from M_0 to M_1 exists. This must be enforced during initial simplification using, for example, the link condition [DEGN98]. Again, we consider a potential swap (v,a,b). There is a homeomorphism between M_1 and the approximation after the swap if the shared boundary between the resulting cluster C_b and each of its neighbors is a single connected chain. Thus, if a swap (v,a,b) creates a new boundary between C_b and any cluster C_c to which it was already adjacent, then the swap does not preserve the homeomorphism and should not be performed.

To see that this is sufficient, consider the effect of swaps on the mesh edges of M_1. Each swap corresponds to zero or more edge-flips in M_1. When two clusters are adjacent on M_0, there is an edge between them in M_1. A swap (v,a,b) may cause, for example, an edge (a,d) to flip to (b,c). Such edge-flipping can never alter the topology of the surface *unless* the edge (b,c) already exists in M_1, in which case the situation is degenerate (see Figure 6). This is exactly the situation that our homeomorphism preserving rule prevents.

The benefit of a swap is how much the error decreases when the swap is performed. For the QEM, we can more efficiently estimate the benefit without actually performing the swap. By assuming fixed vertex positions, the difference between the error before (E) and after (E') moving vertex v from cluster C_a to cluster C_b would simply be

$$E' - E = Q_v(b) - Q_v(a). \tag{1}$$

Note that this is only an estimate, because the vertex positions are not fixed. It is, however, a conservative estimate, as the error can only decrease when computing the new optimal vertex positions from the updated quadrics.

When simplifying texture mapped surfaces, we also add in a second error term that penalizes triangle fold-over by analyzing face normal variance around each vertex. We compute texture coordinates using attribute quadrics [Hop99].

After performing a swap (v,a,b) the new quadrics of a and b are given by $Q'_a = Q_a - Q_v$, $Q'_b = Q_b + Q_v$. We then compute the new optimal vertex positions as before. The connectivity of M_1 may also change, and can be determined as in the initial construction (§3).

The simplest swapping algorithm would pick the highest benefit valid potential swap, perform the swap, and repeat. When no beneficial, valid swaps remain, the algorithm stops. To increase efficiency, we perform an entire independent set of swaps on each pass.

4.3. Coarse-to-Fine Hierarchical Improvement

Our method can improve an entire hierarchy, not just a single approximation. We perform hierarchichal improvement in a coarse-to-fine fashion. Note that when doing hierarchical improvement, the validity rules must be applied with respect to the clustering at the current level and all coarser levels (it is possible for a swap to be valid with respect to one clustering, but not a coarser clustering).

Since we are interested in the quality of the approximation at every level, the single-level QEM is insufficient. When reclustering deep in the hierarchy, we do not want to destroy improvements already made at coarser levels. Thus, we use a hierarchical version of the QEM that is a weighted sum of the quadric errors at each coarser level. When reclustering level k, the aggregate error of level k and all coarser levels is given by $E = \sum_{i=k+1}^{n} (w_i \sum_{u \in M_i} E_u)$. The weights, w_i, are chosen to make the contributions of each level uniform (i.e. to counteract the natural exponential growth of the quadric metric as the number of vertices decreases). This can be achieved by setting

$$\begin{aligned} w_k &= 1 \\ w_{i+1} &= w_i \left(\frac{|M_{i+1}|}{|M_i|} \right)^{\beta}, \end{aligned} \tag{2}$$

where $|M_i|$ denotes the vertex count of level i, and β is a constant corresponding to the growth factor of the error metric. We have found that $\beta \approx 1.9$ for the QEM on many

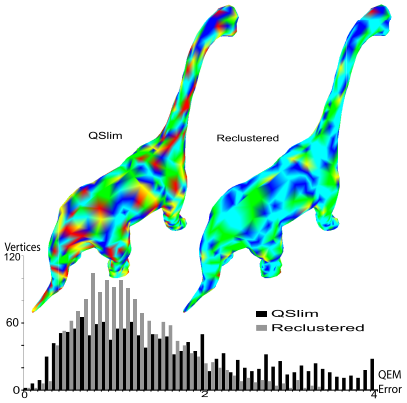


Figure 7: *Improving a static approximation. Warmer colors indicate higher quadric error. As seen in the histogram, our result has more vertices with low error, and fewer with high error.*

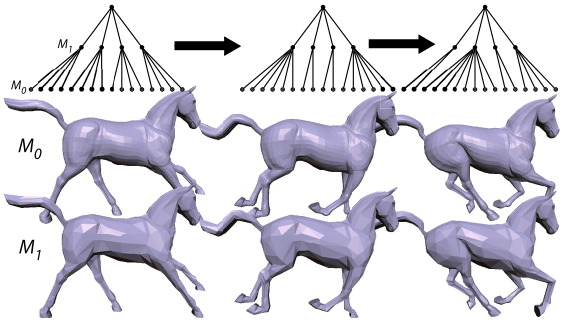


Figure 8: *Starting from an initial hierarchy, we obtain improved hierachies for each subsequent frame, via our reclustering algorithm.*

meshes. All of the examples in this paper were generated using $\beta = 1.9127$, found by regression fitting.

The benefit of a swap (v,a,b) at level k can be computed by applying equation 1 at each coarser level above k (first to a and b, then their parents, grandparents, etc...). Each of these contributions would also be weighted by w_i.

To demonstrate that the improvement process lowers approximation error, an example of improving a two-level hierarchy is shown in Figure 7 where a 23,984 vertex model is approximated by 1,600 vertices. Note that our result has generally lower, and more uniform, error than the original QSlim output. The metric used for the improvement process was the uniform Quadric Error Metric, which was reduced by 26%. The symmetric RMS error was reduced by 5.9% (as measured by MESH [ASCE02]). This discrepency is due to the fact that the QEM error is not completely correlated with RMS error. Although the improvement in this static case is arguably quite modest, the real aim of our method is to handle multiresolution deforming surfaces, as described next.

5. Deforming Meshes

Given a sequence $S = \{S_0, S_1, S_2, ...\}$ of finest-level meshes, all with the same connectivity, we seek to generate a corresponding sequence of hierarchies $H = \{H_0, H_1, H_2, ...\}$ that well represent their respective surfaces. First, we construct H_0 from S_0. This hierarchy can be applied to S_1 as well (*i.e.* using the same contraction clusters). However, since the vertex positions of the finest level mesh have changed, the quadrics will change, and hence the error also changes (generally, it will increase). By running our reclustering algorithm, we adapt the initial hierarchy to reduce the error and better approximate S_1, producing H_1. Continuing in this fashion, H_{i+1} is generated from H_i by improving its error with respect to S_{i+1}. This process is illustrated schematically in Figure 8.

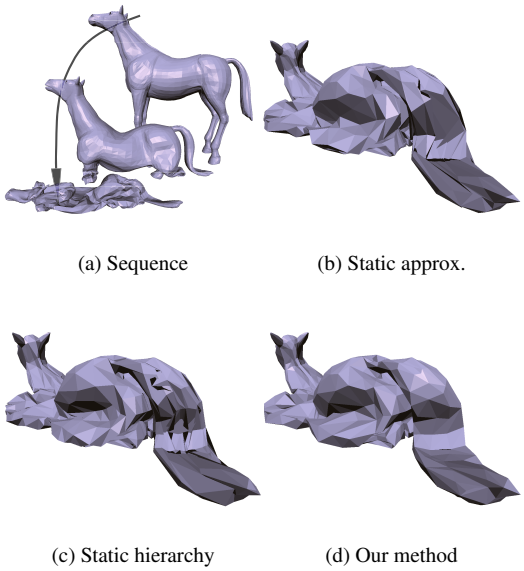
Not only does this process produce a better hierachy for each frame, it also gives us an explicit transformation from H_i to H_{i+1}; namely, the sequence of swaps used to transform one hierarchy into the other. H_0, the vertex positions for each frame, and the sequence of swaps for each frame is all that is needed to encode an entire dynamic multiresolution representation of the deformation. This is our progressive representation, which, even without compression, uses significantly less space than storing full hierarchies for each frame (allowing, for example, low bitrate progressive transmission). Note also that once the swaps have been precomputed in this fashion, the intralevel connectivity of the multilevel mesh can be discarded, leaving what is essentially a vertex hierarchy (which can hence be immediately used in view-dependent refinement and other vertex hierarchy applications).

5.1. Fast Updating

Performing all the swaps to proceed from one frame to the next is somewhat inefficient, mainly because of the need to update the connectivity of the approximation (which can be any arbitrary view-dependent or view-independent cut). However, the effect of a sequence of swaps on the connectivity can also be precomputed.

In the usual binary vertex hierarchies [Hop97], each node has associated with it the faces that will be destroyed or created when the corresponding edge collapse or vertex split occurs. Similarly, viewing the vertices of the multilevel mesh as contraction operation nodes, we associate a *degenerate set* with each node, identifying the finest-mesh triangles that degenerate when the node's children are contracted together.

Given H_i and H_{i+1}, we compute the degenerate sets of each node in H_{i+1}, and store the difference from the corresponding set of H_i. This difference information (plus the sequence



(a) Sequence (b) Static approx.

(c) Static hierarchy (d) Our method

Figure 9: *A horse collapsing like a rubber sheet (original: 8,431v; approximations: 1,600v). The first frame approximation (b) cannot represent later frames well. A "best-cut" approximation from a static hierarchy (c) still suffers from serious artifacts. Our adaptively improved hierarchy (d) yeilds a much nicer approximation.*

of swaps transforming H_i into H_{i+1}) is all the information needed to efficiently replicate the update later.

At playback time, the swaps and degenerate set updates will be performed. Then, assuming there is some arbitrary cut through the tree that should be maintained, the currently active set of triangles will need to be updated. Each triangle that was removed from the degenerate set of a currently contracted node, and not added back into the degenerate set of some other contracted node, must be added to the active set. Likewise, any triangle that was added to the degenerate set of a contracted node, and not originally removed from some other contracted node, must be removed from the active set.

6. Results

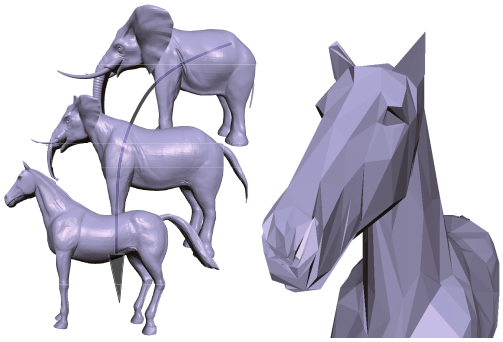
We will now present several results from our system. We begin with an extreme deformation example (Figure 9). In this sequence, a horse model collapses as if it were constructed of a thin rubber sheet. Clearly, any single coarse approximation of this model is likely to be extremely bad for some frame (see Figure 9(b)), thus necessitating a dynamic representation. One possible approach to providing such a dynamic mesh is to still use a static simplification hierarchy, but extract from it a different cut for each frame (based on the quadric error, for example). However, even this yields unpleasent artifacts, as can be seen especially in the tail of the horse. Our method, which quickly provides an adapted hierarchy for each frame, produces a far superior approximation. The three approximations shown in this figure all have 1,600 vertices.

The horse to man morphing sequence (Figures 1 & 14) illustrates the connectivity changes that occur to better suit the hierarchy to the deforming surface. Figure 14 shows several levels detail from the hierarchy. Notice the mesh changes taking place (especially in the chest and shoulder area). Figure 1 shows a closeup of the face in the last frame and a comparison with the first-frame static approximation result. Notice that in the static result the eyes are essentially gone, the mesh itself is extremely ugly, and there are obvious surface inversions (the black triangles). All of these artifacts are *not* present in our result, which additionally has a 34% lower RMS error and an 84% lower QEM error.

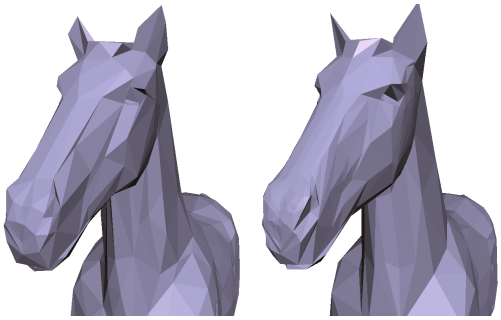
An elephant to horse morphing sequence is shown in Figure 10. The original mesh has 42,900 vertices, the approximations shown each have 800 vertices. Notice that the nose of the horse is terribly malformed in the static hierarchy case. Our dynamic hierarchy does a significantly better job on the nose and eyes. For comparison, we show also the result of applying QSlim directly to the last frame of the sequence.

Figure 11 shows an RMS error (as measured by Metro [CRS98]) comparison of our method with several alternatives. This graph was generated for an 800 vertex approximation of a galloping horse animation (Figure 8). The First-frame Static method is using the approximation from the first frame for all subsequent frames (only its vertex positions change). The Direct QSlim method is generating an entirely new approximation each frame, without regard to history. The "Average" Mesh method modifies the QSlim algorithm to produce a single (static connectivity) approximation based on all frames. This is done by collecting the edge-collapse of least cost from each frame, and selecting among these the edge-collapse whose maximum cost over all frames is smallest. This edge-collapse is then performed for every frame, and the process repeats. From the graph, it is clear that our method generally results in smaller RMS errors than the two static connectivity methods. We can also see that there is no appreciable accumulation of error. Other experiments on longer and more extreme deformation cases also showed no noticable error accumulation. Finally, even though our error is higher than that of an entirely new approximation per frame, the lack of temporal coherence in such a method introduces such obvious, terrible flickering that it is rendered useless (see the accompanying video).

Since our multilevel mesh is, in some sense, an extension of vertex hierarchies, it can be easily used in existing adaptive refinement schemes. Figure 12 demonstrates this with a galloping elephant, refined adaptively along one of its axes. Once a sequence of swaps has been precomputed for the animation, it can be progressively played back as the animation



(a) Sequence (b) Static hierarchy



(c) Direct QSlim (d) Our method

Figure 10: *Elephant to horse morph (original: 42,900v; approximations: 800v). (b) "Best-cut" from a static hierarchy. (d) Our dynamic hierarchy better approximates highly deformed states. (c) The direct QSlim approximation, for comparison.*

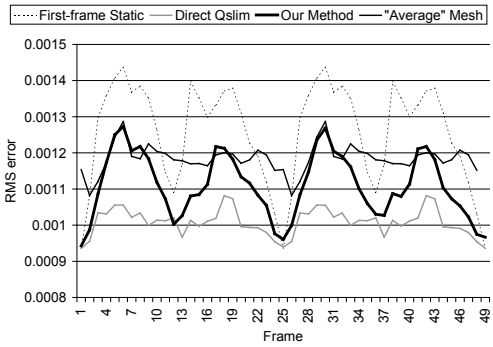


Figure 11: *Various methods' RMS error per frame for an 800 vertex approximation of a galloping horse.*

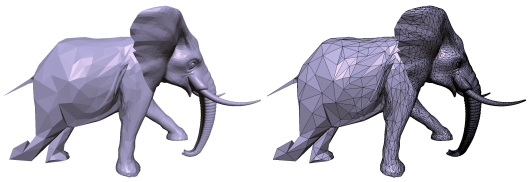


Figure 12: *The multilevel mesh structure can easily be used with adaptive refinement schemes. Here, refinement is based on position along the x axis.*

proceeds. Our progressive representation uses significantly less space than storing an entire hierarchy on each frame, and provides a format suitable for streaming animation applications. An adaptive cut, such as the one shown in the figure, can be efficiently maintained even as the hierarchy changes.

Our method is efficient in terms of both time and space. All times reported here were measured on a 1.7Ghz Pentium IV with 512MB RAM. The pre-processing (reclustering) stage takes an average of 4.3 seconds per frame to generate the swap sequence for a hierarchy of the collapsing horse with a branching factor of eight. This is comparable to the cost of rebuilding the hierarchy with the QSlim algorithm, approximately 6.9 seconds per frame. The pre-processing time is, however, highly dependent on the branching factor (smaller branching factors produce more levels, and each additional level incurs measurable overhead). Storing the swaps (without any additional compression) for the collapsing horse sequence requires about 2.2KB per frame (using two 32-bit integers to identify each swap). Storing full degenerate set update information, plus swaps, requires about 18.5KB per frame. By comparison, storing the full hierarchy (a single integer for each node indicating who its parent is, plus the degenerate sets for each node) would require 103.4KB per frame. These numbers do not include the cost of storing the vertex positions (which is substantial, but could be compressed using existing methods [AM00]).

The run-time costs depend, of course, on the degree of deformation per frame, and the size of the input mesh, but per-frame time to update the hierarchy is typically in the 2ms to 12ms range for moderate size input meshes (around 10,000 vertices). Figure 13 shows hierarchy update times for each frame of the collapsing horse and galloping horse. Note that the collapsing horse (which has several frames of relatively high cost) represents essentially a "worst case" scenario, since it contains drastic deformation over a very short time period. For increased run-time performance, less important swaps (i.e. ones that do not improve the metric appreciably) could be discarded during pre-computation.

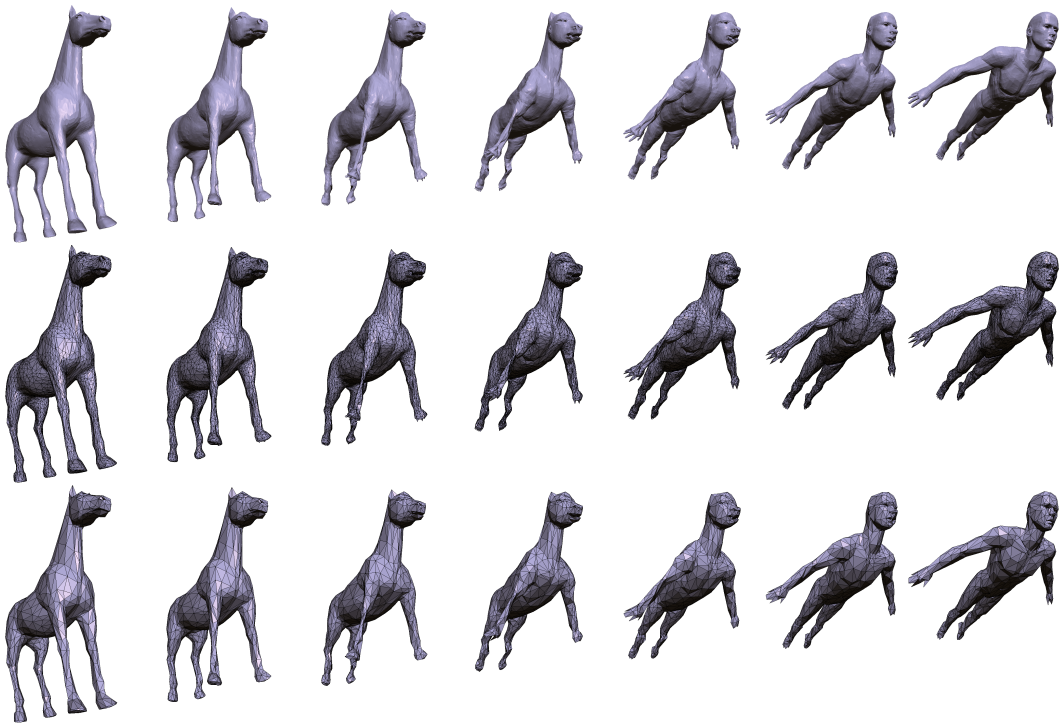


Figure 14: *Levels of detail (original, 3200v, 800v) from the horse-to-man multilevel mesh sequence. Each approximation level adapts over time.*

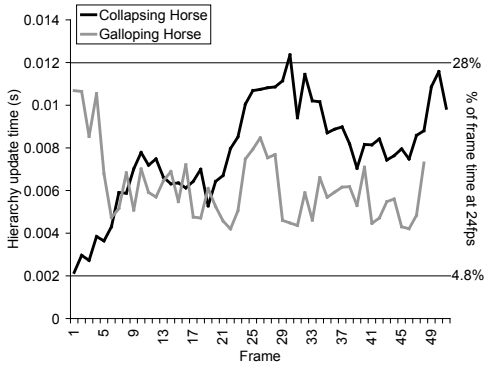


Figure 13: *Run-time hierarchy update cost per frame of the collapsing horse and galloping horse sequences.*

7. Conclusion & Future Work

We have presented a multi-resolution representation suitable for deforming meshes that allows incremental improvements to the hierarchy via reclustering. By storing mesh connectivity at each level of the hierarchy, we can guarantee that only valid updates are performed. Our representation also enables the induced connectivity changes to be progressively played back in real-time, providing an encoding suitable for animation. Once the hierarchy has been updated to the desired frame, various approximations can be extracted from it in an adaptive manner.

Our technique works quite well; however, there are a number of intersting directions for future research stemming from our work. Currently, we do not allow clusters to evaporate completely, because it would complicate hierarchy maintenance. This can result in the algorithm getting "stuck" unecessarily in a constrained local minimum. Removing this restriction is an avenue that will be investigated. Also, our progressive representation of connectivity changes over time has the potential to become the basis for a compression scheme for non-static connectivity, since the degenerate set differences can be thought of as connectivity residuals. In combination with a vertex-based animation compression scheme, such as [AM00], it may be possible to produce an effective compression scheme for an entire multiresolution representation of a deforming mesh, without completely throwing away its connectivity as is done in geometry videos [BSM*03].

Acknowledgements We wish to thank Robert W. Sumner and Jovan Popović for providing the gallpoing horse, galloping elephant, and collapsing horse animations. We also thank

Alla Sheffer for providing the horse-to-man and elephant-to-horse morphing correspondence data.

References

[AL02] AHN M., LEE S.: Mesh metamorphosis with topology transformations. In *Proc. 10th Pacific Conference on Computer Graphics and Applications* (2002), IEEE Computer Society, p. 481.

[ALS04] AHN M., LEE S., SEIDEL H.-P.: Connectivity transformations for mesh metamorphosis. In *Eurographics Symposium on Geometry Processing 2004* (2004), The Eurographics Association, pp. 77–83.

[AM00] ALEXA M., MULLER W.: Representing animations by principal components. In *Eurographics* (2000), The Eurographics Association/Blackwell Publishers.

[ASCE02] ASPERT N., SANTA-CRUZ D., EBRAHIMI T.: Mesh: Measuring errors between surfaces using the hausdorff distance. In *Proc. IEEE International Conference on Multimedia and Expo* (2002), vol. I, pp. 705 – 708. http://mesh.berlios.de/.

[BSM*03] BRICENO H. M., SANDER P. V., MCMILLAN L., GORTLER S., HOPPE H.: Geometry videos: a new representation for 3d animations. In *Proc. 2003 ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2003), Eurographics Association, pp. 136–146.

[CH04] CARR N. A., HART J. C.: Two algorithms for fast reclustering of dynamic meshed surfaces. In *Eurographics Symposium on Geometry Processing 2004* (2004), The Eurographics Association, pp. 229–239.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[DEGN98] DEY T., EDELSBRUNNER H., GUHA S., NEKHAYEV D.: Topology preserving edge contraction, 1998. Technical Report RGI-Tech-98-018, Raindrop Geomagic Inc., Research Triangle Park, North Carolina.

[DR05] DECORO C., RUSINKIEWICZ S.: Pose-independent simplification of articulated meshes. In *Symposium on Interactive 3D Graphics* (Apr. 2005).

[Gar99] GARLAND M.: Multiresolution modeling: Survey & future opportunities, 1999. In *Eurographics '99 – State of the Art Reports, pages 111–131, 1999.*

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[Hop96] HOPPE H.: Progressive meshes. In *SIGGRAPH '96* (1996), ACM Press, pp. 99–108.

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *SIGGRAPH '97* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 189–198.

[Hop99] HOPPE H.: New quadric metric for simplifying meshes with appearance attributes. In *Proc. 10th IEEE Visualization Conference* (Washington, DC, USA, 1999), IEEE Computer Society.

[KK98a] KARYPIS G., KUMAR V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput. 20*, 1 (1998), 359–392.

[KK98b] KARYPIS G., KUMAR V.: Multilevel k-way partitioning scheme for irregular graphs. *J. Parallel and Distributed Computing 48*, 1 (1998), 96–129.

[KL70] KERNIGHAN B. W., LIN S.: An efficient heuristic for partitioning graphs. *Bell Systems Tech. J. 49* (Feb. 1970), 291–308.

[LDSS99] LEE A., DOBKIN D., SWELDENS W., SCHRÖDER P.: Multiresolution mesh morphing. In *SIGGRAPH '99* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, pp. 343–350.

[LE97] LUEBKE D., ERIKSON C.: View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH '97* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 199–208.

[LL05] LIN C.-H., LEE T.-Y.: Metamorphosis of 3d polyhedral models using progressive connectivity transformations. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 2–12.

[LRC*02] LUEBKE D., REDDY M., COHEN J. D., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3-D Graphics*. Morgan Kaufmann, 2002.

[MG03] MOHR A., GLEICHER M.: *Deformation Sensitive Decimation*. Tech. rep., University of Wisconsin Graphics Group, 2003. http://www.cs.wisc.edu/graphics/Gallery/DSD/.

[SP01] SHAMIR A., PASCUCCI V.: Temporal and spatial level of details for dynamic meshes. In *Proc. ACM symposium on Virtual reality software and technology* (2001), ACM Press, pp. 77–84.

[SPB00] SHAMIR A., PASCUCCI V., BAJAJ C.: Multiresolution dynamic meshes with arbitrary deformations. In *Proc. Visualization '00* (2000), IEEE Computer Society Press, pp. 423–430.

[SWG*03] SANDER P. V., WOOD Z. J., GORTLER S. J., SNYDER J., HOPPE H.: Multi-chart geometry images. In *Proc. Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), Eurographics Association, pp. 146–155.

[XV96] XIA J. C., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. In *Proc. 7th conference on Visualization* (1996), IEEE Computer Society Press, pp. 327–ff.