Simplification and Real-time Smooth Transitions of Articulated Meshes

Jocelyn Houle Pierre Poulin

Départment d'informatique et de recherche opérationnelle
Université de Montréal

Abstract

Simplification techniques have mainly been applied on static models. However in movie and game industries, many models are designed to be animated. We extend the progressive mesh technique to handle skeletally-articulated meshes in order to obtain a continuous level-of-detail (CLOD) representation that retains its ability to be animated. Our technique is not limited to any simplification metric, nor is it limited to generating models composed of a subset of the original vertices. It thus preserves the full simplification potential.

To further improve performance, we can use this CLOD representation and extract a discrete set of skeletally-articulated models. Each model can be independently optimized, such as by using triangle strips. We can also use morphing between the different models in order to create smoother transitions. The result is a more accurate representation of animated articulated models, suitable for real-time applications.

Key words: Mesh simplification, skeletal animation, skinning, LOD, real-time rendering.

1 Introduction

Polygon throughput is ever-increasing. However, even though graphics hardware accelerators have greatly improved in the past few years, pushing the limit does not remove it altogether: we will always need to handle too big a scene.

Often, parts of the scene feature more details than necessary. In fact, a model, depending on its rendering characteristics, whether it be screen size, illumination, visibility, or movement, does not always need to be manipulated at full complexity.

One solution is to manually build multiple versions of the same model, and then select the appropriate LOD to display. This technique was common in early flight simulators where memory and speed where essential, and is now extensively used in games for roughly the same reasons. Coming up with a set of LODs is a time-consuming process, requiring lots of meticulous work by an artist. Since the LODs are different, switching between them introduces popping which unfortunately helps noticing the transition. Considerable time and effort must be spent by the artist in order to further address such issue.

Several common automatic techniques (adaptive subdivision of parametric surfaces, subdivision surfaces, etc.) successfully reduce a model's complexity while trying to preserve most of its appearance. Automatic techniques greatly help reduce the modeling effort needed to generate simplified models. One such family of techniques, mesh simplification, is applied to polygonal models.

Edge collapse simplification operates by repeatedly merging pairs of adjacent vertices. The progressive mesh technique by Hoppe [7, 8] constitutes a common foundation for such techniques. According to a metric, a cost of collapse is associated to every edge in a mesh. We can then iteratively remove the lowest-cost edge and update the mesh accordingly. By keeping the inverse information, a vertex split (or *vsplit*), we can effectively reverse the simplification process. We end up with a multiresolution representation, composed of a base model and an associated ordered list of *vsplits*, which can be coarsened or refined at will.

Many more simplification techniques exist, and the reader is directed towards more exhaustive surveys [2, 16, 1, 3].

2 Previous Work

This section addresses only work related to articulated mesh simplification.

In a skeletally-articulated mesh, when a bone moves, its associated vertices follow its transformation. These meshes are a common way of modeling moving characters in current animation software. Real-time applications also tend to use articulated models in order to save memory and exploit new features such as frame interpolation and inverse kinematics. To date, we are aware of only two simplification techniques that handle articulated meshes.

Schmalstieg and Fuhrmann [17] decompose the model into regions corresponding to the model's bones. Vertices shared (weighted) by a common set of bones form additional regions. Edges are collapsed only between vertices that lie within the same region. Therefore, this technique effectively simplifies as many submodels as

there are regions; adjacent regions affect one another only when boundary faces collapse.

The second technique, from Huebner [9], is closely related to Hoppe's progressive mesh representation [7]. The model's vertices are placed in order of creation during the refinement process (i.e., when applying the *vsplits*). Since edges are constrained to collapse only on either endpoint, the complete vertex array of the detailed mesh can be used for all the generated LODs: lower resolution models simply use an early portion of that array. This approach allows the vertices to be resident in video memory, and leaves only the face management to the CPU. Special considerations for texture seams, attribute conservation, and pose choice have also been described. The resulting technique offers a multi-resolution representation manageable in real-time with current graphics APIs. It has has been successfully used in a commercial 3D game [10].

Both simplification techniques use only a subset of the original vertices (this is sometimes referred to half-edge collapse simplification [11]). An advantage of such an approach is that all the vertices stay valid in the simplified mesh, whereas arbitrarily moving a vertex might cause unintended results due to the skeletal transforms. Unfortunately, such an approach also forbids any vertex position optimization during the simplification process. This potentially reduces the quality of the generated simplified mesh. Also, these techniques cannot benefit from face encodings such as triangle strips because polygons can be removed anywhere in the final representation.

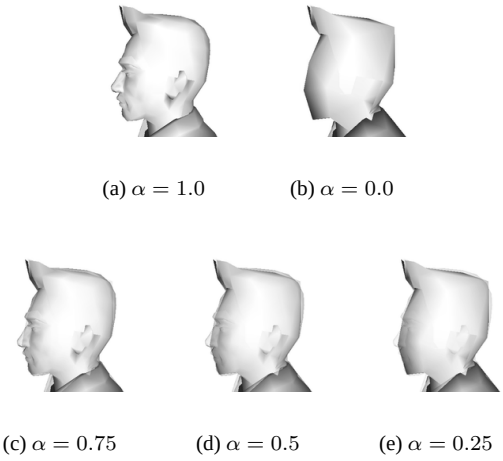
Our technique offers the same major benefit as these techniques (notably, being able to animate the simplified mesh), but can also help with rendering performance and LOD transition quality.

3 Overview

Starting with a skeletal mesh, our method effectively constructs a multi-resolution representation. This articulated progressive mesh can be coarsened or refined at will, for any pose of the articulated model. In fact, it can be used with any animation sequence applicable to the base mesh. Although our implementation yielded interactive results (a few frames a second), this representation proved insufficient for real-time applications where multiple models would be used simultaneously.

However, this automatic simplification technique can construct any number of discrete articulated LODs. These LODs are in a form suitable to optimize rendering, such as by building strips, desirable in real-time applications.

The transition between two LODs usually exhibits noticeable popping. A typical way of reducing this popping is by alpha-blending the two LODs together. Unfortunately, alpha-blending simply makes features gradually appear or disappear, which can look awkward, especially on the silhouette (see Figure 1, as well as accompanying animated sequences, for examples). We invite the reader to visit the web site associated with this paper from *http://www.iro.umontreal.ca/labs/infographie/papers* to visualize the various animations referenced throughout this paper.



(a) $\alpha = 1.0$ (b) $\alpha = 0.0$

(c) $\alpha = 0.75$ (d) $\alpha = 0.5$ (e) $\alpha = 0.25$

Figure 1: Example of transition using alpha-blending. Figures (a) and (b) illustrate the two transition LODs. Figures (c) to (e) show the actual alpha-blending results.

Our technique instead constructs correspondences between vertices and attributes, enabling mesh morphing between the articulated LODs. Such mesh morphing, being more semantically accurate, considerably enhances transitions.

4 Simplification

Our simplification technique proceeds as follows. The detailed articulated base mesh is transformed into a static mesh. We simplify this static detailed model, essentially constructing a progressive mesh. Because simplification is applied to a static mesh, we can use any edge collapse simplification metric. Our implementation used a memoryless quadric error metric [4, 5, 13, 14, 6] applicable to non-manifold models. We finally reinterpret the *vsplit* structures into the skeleton in order to construct an articulated progressive mesh.

The main idea of our technique is that rather than being applied directly on the articulated model, the preprocess stage of our simplification is applied to a static pose of the model. We simply copy the *vsplits* into the articulated

progressive mesh and interpret them with respect to the skeleton. The simplified articulated model coincides with the static progressive mesh when its pose is the same as the one used during the static model simplification.

4.1 Back-propagation

Since a vertex can move after an edge collapse, we must propagate this translation from world-space (computed during the simplification) back to bone-space (the correct space in the skeletal mesh). In order to do this, we precompute the inverse of every bone transformation that we compose using the skeletal hierarchy. The vertex translation $\Delta\mathbf{P}$ is then transformed into the proper bone-space before being applied.

4.2 Weighted Vertices

Articulated meshes often use weighted vertices in order to smooth bending at the joints. A weighted vertex with multiple representations is expressed as:

$$\mathbf{P} = \sum_{i=1}^{n} \mathbf{P}_i w_i \qquad \text{with} \quad \sum_{i=1}^{n} w_i = 1 \tag{1}$$

where n is the number of representations (usually, $n = 2$), each described with a position $\mathbf{P}_i$ (in world-space) and an associated weight w_i. Other skinning techniques exist, like the recent work by Lewis *et al.* [12], but none are as widespread as the weighted vertices technique.

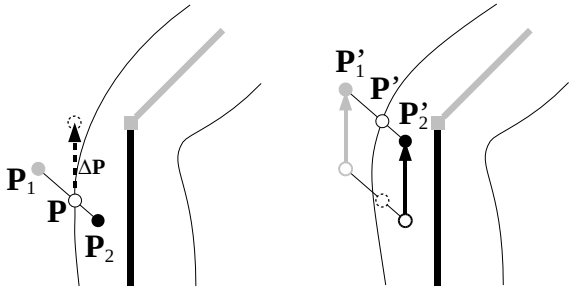
In order to obtain a correct weighted vertex position, we must modify its weighted representations. Applying the translation $\Delta\mathbf{P}$ to every $\mathbf{P}_i$ effectively translates the weighted position $\mathbf{P}$ to the correct final position $\mathbf{P}'$:

$$\begin{aligned} \sum_{i=1}^{n} (\mathbf{P}_i + \Delta\mathbf{P}) w_i &= \sum_{i=1}^{n} \mathbf{P}_i w_i + \Delta\mathbf{P} \sum_{i=1}^{n} w_i \\ &= \mathbf{P} + \Delta\mathbf{P} \\ &= \mathbf{P}'. \end{aligned} \tag{2}$$

Figure 2 illustrates this property. We could have used other schemes (namely, any transformation which leads to a final interpolation into $\mathbf{P}'$), but this one was found appropriate, being simple and intuitive.

As with vertices, other weighted attributes (notably, normals) undergo similar modifications based on the bones to which they are related.

We have not addressed how new weights are assigned on the resulting vertex of an edge collapse: the inital values are merely carried throughout the simplification process. This proved satisfactory in our context. A more thorough approach would try to reassign more appropriate weights to the displaced vertices. This would need to consider the initial weight assignment function, or a close approximation.[1]



(a) Translating the interpolated vertex.

(b) Interpolating the translated weighted representations.

Figure 2: Consider $\boldsymbol{P}$ a vertex defined by two weighted representations $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, each associated with a bone (the association is illustrated with different shades). Following an edge collapse, assume the resulting vertex is translated by $\Delta\boldsymbol{P}$ in (a). We can observe in (b) that translating every weighted representation by the same vector yields the correct interpolated position $\boldsymbol{P} + \Delta\boldsymbol{P} = \boldsymbol{P}'$.

4.3 Results

Our resulting method can be applied to any skeletally-animated model, and offers a CLOD representation with interactive performances. Figure 3 shows various LODs taken from such a representation. The original model, containing 4308 faces, took 623 ms to simplify on an Athlon 600 MHz. As mentioned earlier, the metric used is a memoryless constrained quadric error metric. We also have accompanying animated sequences from the web site that show various LODs generated by our technique.

Our simplification process only took into account a single pose out of all the possible poses of the model. Since the choice of this pose can affect the faces and vertex positions of the model, it must be appropriately selected. In order to reduce the deviation on the other frames, we used a pose in which the members are half-bent. Using members in a straight position would have increased the deformation deviation of weighted vertices when the member is fully flexed. In essence, we are splitting this error, albeit very small, in half. This approach proved to be quite sufficient in practice.

We could easily consider a weighted combination of

[1] Our approach can be considered such an approximation for small translations.

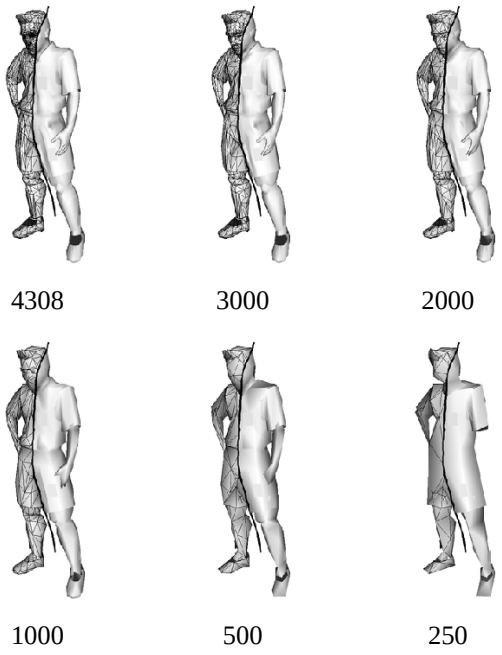


Figure 3: Simplification results, with associated face counts.

the edge collapse costs in all the poses of an animated sequence, or in a set of extreme poses. This set of poses should preserve the more dynamic parts of the model longer without unduly increasing processing time. Such a multi-pose simplification surely helps in choosing a best edge to collapse, but unfortunately increases the difficulty of defining a best position of collapse.

The simplified model is an articulated CLOD representation that can easily be managed interactively. Unfortunately, it is not quite fast enough to manage multiple models simultaneously. To do so, we must lower the runtime requirements.

5 Transition by Morphing

In order to improve the rendering speed, we construct discrete LODs. This approach has been used with static models for quite some time. However, the transition between two LODs usually exhibits popping, due to abrupt differences in visual appearance. We exploit our CLOD representation similarly to Hoppe's geomorphs [7] in order to improve the transition between two such LODs.

5.1 Vertex Positions

The *vsplit* structures indicate towards which vertex another collapses. This correspondence can be chained with further collapses involving the same vertex. We therefore know, for each vertex of the detailed mesh M_d, towards which vertex in the simplified mesh M_s it will ultimately end up. By linearly interpolating the positions, we construct a geomorph from M_d to M_s.

During the geomorph, faces exclusive to M_d gradually degenerate until the remaining faces exactly match those of M_s. We must also interpolate the face attributes for the degenerate M_d to appear exactly as M_s. Compared to alpha-blending, such a morphing offers more semantically accurate transitions.

Morphing is controlled by a parameter t varying from 0 (which corresponds to M_d) to 1 (which corresponds to M_s). Vertex interpolation can be expressed as follows:

$$\mathbf{P}'_i = \mathbf{P}_i(1-t) + \mathbf{P}_{f(i)}t \qquad 0 < t < 1 \qquad (3)$$

where $\mathbf{P}_i$ is the original vertex position in M_d, $\mathbf{P}_{f(i)}$ is the corresponding final position in M_s, and $f(i)$ is the correspondence function. Such a formulation is quite fast in practice since we start with a copy of M_d, which already contains $\mathbf{P}_i$.

5.2 Attributes

Interpolation of normals is done in a similar fashion. Although quaternions provide the correct interpolation between two normals, a possible shortcut, albeit incorrect, consists in interpolating linearly the two tips of the normals (with or even without renormalization). Since morphing is typically done in less than a second, we observed no objectionable artifacts. Transition smoothness appears thus more important than transition correctness.

Although our current implementation only manages vertices and normals, further attributes (vertex color, texture coordinates, etc.) could be integrated as well. Since the morphing is done swiftly, such attributes most probably can get away with similarly incorrect interpolation unnoticed.

5.3 Approach

We have chosen to apply the morphing interpolation on static meshes instead of articulated ones, mainly for performance issues. Since articulated meshes typically contain a number of weighted vertices, interpolation of such vertices would have to be applied to every vertex representation. This would needlessly multiply the processing requirements since the morphing results are the same.

Our final structure consists in a series of m discrete articulated LODs combined with a series of $m-1$ correspondence tables describing morphing between every two consecutive LODs. These correspondences (simple index pairs) take litte extra memory compared to the complete meshes. Consecutive morphing correspondences can be chained to allow morphing of non-consecutive LODs.

To avoid always morphing a model, we can construct 3D morphing zones (or slices) based on distance to the

camera. These zones, which can be suitably defined by the user, are separated by appropriately larger non-morphing zones where the discrete (optimized) skeletal LODs are used. Such an approach can amortize the overhead associated with the morphing by restraining the region of morphing, and therefore, by reducing the overall time spent morphing.

5.4 Results

Our approach builds two static LODs based on skeletal ones, and interpolates between them. Since transforming a skeletal model into a static one is usually a well optimized path (real-time applications typically do so each time they render the mesh), we consider this operation to incur only reasonable overhead.

The rendering time of the morphed model is the same as that of the highest-resolution LOD since it has exactly the same number of faces, vertices, and attributes: only the actual values are different.

The morphing time directly depends on the number of elements to morph, which is proportional to the number of different vertices and attributes between two LODs. In our unoptimized implementation capable of more that 500 000 interpolations per second (vertices and normals combined), morphing time is slightly faster than rendering time of the resulting static mesh. This indicates that morphing slows down the performance by half, but it is so only for the actual duration of the morphing. Afterwards, greater gains are attained with the lower-resolution model, which can even be optimized with triangle strips.

Compared to alpha-blending two LODs, our technique shares roughly the same processing requirements: both the lower- and higher-resolution models must be transformed prior to rendering. Our technique simply replaces the lower-resolution rendering with the morphing step. We feel the morphing offers far more pleasing transitions for a comparable overall cost. For cases where the whole LOD is resident in video memory, alpha-blending is faster because our technique needs to send the vertices every time. Upcoming vertex shading rendering architectures eliminate that advantage, and permit to do both the skeletal and morphing transforms on the graphics chip specialized hardware. We assume this would further increase the appeal of our technique.

Figure 4 shows a morphing between two animated LODs. The morphing is almost total since less than 5% of the vertices and normals of the mesh are left untouched.

We have found that the morphing needs little time in order to improve transition smoothness. In fact, only a few frames morphing in less than a second are enough to remove popping. An animated sequence from the same web site shows such examples.

6 Conclusion

We have described a simplification technique which can be applied to skeletally-animated meshes. Furthermore, we have described how to use this CLOD representation in order to easily generate discrete skeletal LODs which can smoothly be morphed in real-time.

Our approach allows the various LODs to be optimized for rendering, such as by using strips. Because we must interpolate the vertices every frame, we cannot take advantage of hardware support for skeletal meshes. However, the simplicity of our technique makes it a good candidate for a hardware implementation on future architectures.

Further research is warranted, especially in terms of quality evaluation of the simplified models. A metric integrating the rendering impact of the simplification and morphing (inspired by recent image-based metrics [15]) could probably improve even more LOD transition, as well as offer a heuristic for automatic LOD selection. Optimal vertex placement, with consideration to weights, is also an avenue left unexplored. A tight integration with current and next generation graphics hardware is also intended.

7 Acknowledgments

This research was made in collaboration with Electronic Arts Canada, under the supervision of John W. Buchanan. The authors would like to thank Alain Fournier, and people from IMAGER at UBC and LIGUM at UdeM for many enlightening discussions. The models and animation sequences are courtesy of Electronic Arts. This research was made possible by grants from Electronic Arts Canada and NSERC.

References

[1] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, February 1998.

[2] Carl Erikson. Polygonal simplification: An overview. Technical Report TR96-016, University of North Carolina, Department of Computer Science, 1996.

[3] Michael Garland. Multiresolution modeling: Survey & future opportunities. State of the art report (STAR), Eurographics '99, 1999.

[4] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 209–216, August 1997.

[5] Michael Garland and Paul S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Visualization '98 Proceedings*, pages 263–269, October 1998.

[6] Hughes Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *IEEE Visualization 1999*, pages 59–66, October 1999.

[7] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series, pages 99–108, August 1996.

[8] Hugues Hoppe. Efficient implementation of progressive meshes. *Computers & Graphics*, 22(1):27–36, February 1998.

[9] Robert Huebner. Application of continuous level-of-detail for 3D games. In *SIGGRAPH 2000 Course Notes*, number 39, July 2000.

[10] Robert Huebner. Bringing life to the undead. In *Game Developer's Conference 2000*. http://www.nihilistic.com/GDC2000/gdc2000lecture, 2000.

[11] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. A general framework for mesh decimation. In *Graphics Interface '98*, pages 43–50, June 1998.

[12] J.P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, pages 165–172, July 2000.

[13] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *Proceedings of IEEE Visualization '98*, pages 279–286, October 1998.

[14] Peter Lindstrom and Greg Turk. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):98–115, April – June 1999.

[15] Peter Lindstrom and Greg Turk. Image-driven simplification. *ACM Transactions on Graphics*, 19(3):204–241, July 2000.

[16] David Luebke. A survey of polygonal simplification algorithms. Technical Report TR97-045, University of North Carolina, Department of Computer Science, December 1997.

[17] Dieter Schmalstieg and Anton Fuhrmann. Coarse view-dependent levels of detail for hierarchical and deformable models. Technical Report TR-186-2-99-20, Vienna University of Technology, 1999.

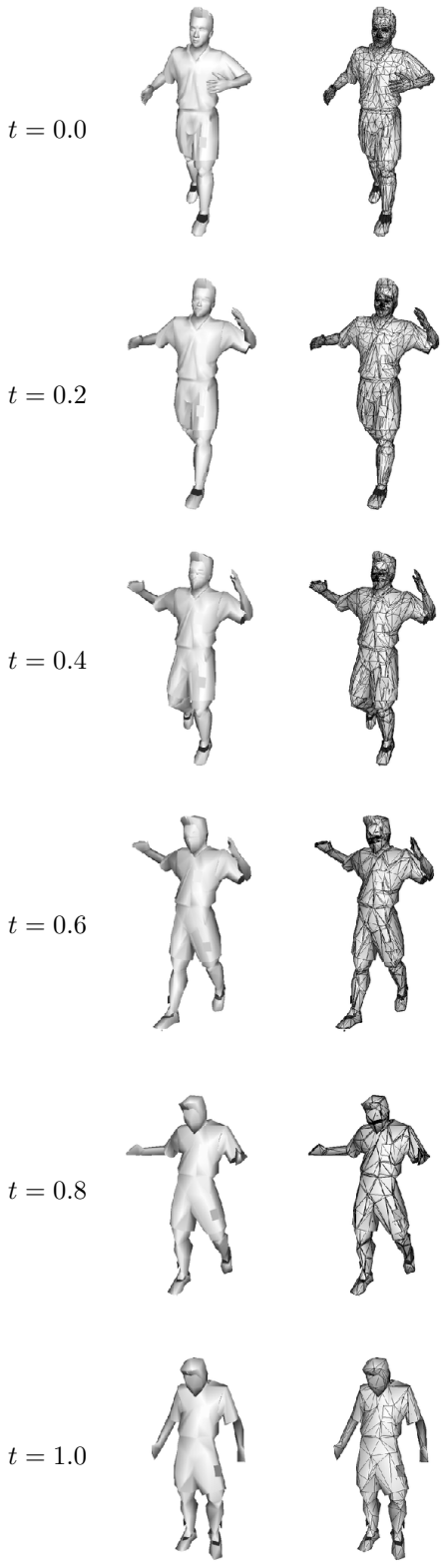


Figure 4: Morphing example: from 4308 to 500 faces.