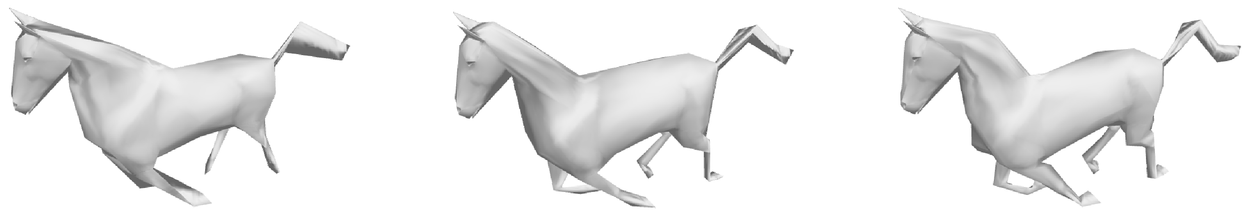
Analytic simplification of animated characters

Bruce Merry[*]
ARM

Patrick Marais[†]
University of Cape Town

James Gain[‡]
University of Cape Town



(a) rest-pose LOD, 337 influences (b) AS LOD, 338 influences (c) inf. simplification, 341 influences

Figure 1: *Three forms of influence simplification with comparable influence counts. (a) Traditional simplification based on the rest pose collapses the knees and ankles. (b) Simplification in animation space, our first contribution, improves the rear ankles and the tail. (c) Influence simplification, our second contribution, further improves the front legs and the curve of the neck and back.*

Abstract

Traditionally, levels of detail (LOD) for animated characters are computed from a single pose. Later techniques refined this approach by considering a set of sample poses and evaluating a more representative error metric. A recent approach to the character animation problem, *animation space*, provides a framework for measuring error analytically. The work presented here uses the animation-space framework to derive two new techniques to improve the quality of LOD approximations.

Firstly, we use an animation-space distance metric within a progressive mesh-based LOD scheme, giving results that are reasonable across a range of poses, without requiring that the pose space be sampled.

Secondly, we simplify individual vertices by reducing the number of bones that influence them, using a constrained least-squares optimisation. This *influence simplification* is combined with the progressive mesh to form a single stream of simplifications. Influence simplification reduces the geometric error by up to an order of magnitude, and allows models to be simplified further than is possible with only a progressive mesh.

Quantitative (geometric error metrics) and qualititative (user perceptual) experiements confirm that these new extensions provide significant improvements in quality over traditional, naïve simplification; and while there is naturally some impact on the speed of the off-line simplification process, it is not prohibitive.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism — Animation; I.3.5 [Computing Methodologies]: Computer Graphics—Computation Geometry and Object Modeling — Geometric algorithms, languages, and systems

Keywords: character animation, simplification

[*]e-mail:bruce.merry@arm.com
[†]e-mail:patrick@cs.uct.ac.za
[‡]e-mail:jgain@cs.uct.ac.za

1 Introduction

Character animation brings a static model to life by defining how its geometry changes over time. In real-time applications (such as games), the standard approach is to model a stick-figure skeleton, or *rig*, then weight the influence of each bone on each vertex of the model, or *skin*. The animator may then control the bones of the rig, and the skin will deform accordingly.

Level of detail (LOD) refers to a class of techniques for managing the complexity of highly detailed models. Multiple representations of the base model are created, with varying amounts of detail, each of which should approximate the original as well as possible. The most appropriate representation is then selected for the task at hand. Standard applications include rendering (with distant models at lower detail), compression, and progressive transmission [Hoppe 1996].

In this paper, we address the problem of combining the fields of character animation and level-of-detail. Traditionally, LOD representations of an animated character have been produced by applying a static LOD algorithm to a single pose of the character, namely the *rest pose* in which the model was created. Since this is only designed to be a good approximation in that pose, the quality in other poses may suffer. Figure 1(a) shows an example of this: the rest pose has the horse standing straight, so the naïve approach eliminates the geometry necessary to represent the ankles and makes a poor approximation to the tail in the pose shown.

Our approach is based on a particular method of character skinning, *animation space*, which represents vertices as points in a high-dimensional space. We measure the error of approximations directly in this space, rather than in any particular pose. Figure 1(b) shows the advantage of this strategy.

Standard LOD techniques work by reducing the geometric detail of models — eliminating vertices, edges and faces. However, the

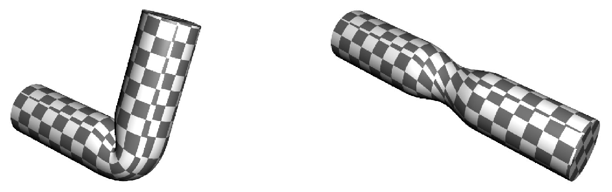


Figure 2: *The collapsing-elbow and candy-wrapper effects are well-known flaws in SSD.*

cost of either storing or rendering a model depends on the number of influences of bones on vertices: in our implementation, a vertex with four influences takes roughly three times as long to transform as a vertex with only one influence. A further contribution is *influence simplification*, an LOD technique that removes influences from vertices, and adjusts the remaining influences to minimise the resultant error. We also demonstrate that influence simplification may be combined with a progressive mesh in a unified framework. Figure 1(c) shows an example of this.

We start with general background on character animation and LOD in Section 2. Section 3 covers previous approaches to merging the two fields. Sections 4 and 5 describe our two contributions, namely progressive meshes in animation space and influence simplification. We finish with results in Section 6 and conclusions in Section 7.

2 Background

2.1 Character animation

There are many algorithms used for character animation; Collins and Hilton [2001] provide a survey. We will describe only two animation methods here: skeletal subspace deformation (SSD) and animation space (AS) [Merry et al. 2006]. Both are real-time skeletal animation methods, meaning that an animator positions the bones of the model, and this in turn drives the skin. The relationship between the bones and the skin is controlled by a set of weights.

2.1.1 Skeletal subspace deformation

In the case of SSD, there is a scalar weight for every bone-vertex pair. Most of these weights are zero, since for example, it makes no sense for the position of a wrist bone to have any influence on the feet. We refer to the non-zero weights as *influences*: the bone is said to influence the vertex.

At this point, we introduce some notation. Let $\mathbf{v}$ be a vertex that is to be animated, w_i be the weight corresponding to this vertex for bone i, and G_i be the matrix that transforms from the local space defined by bone i to model space. Variables marked with a hat ($\hat{G}$ and $\hat{\mathbf{v}}$) indicate values in the rest pose, while variables without a hat refer to the current pose of an animation. Between the rest pose and the current pose, each bone i is transformed by $G_i\hat{G}_i^{-1}$, and SSD transforms $\hat{\mathbf{v}}$ into $\mathbf{v}$ by a weighted linear combination of these transformations:

$$\mathbf{v} = \sum_i w_i G_i \hat{G}_i^{-1}\hat{\mathbf{v}} \quad \text{where } \sum_i w_i = 1. \tag{1}$$

The transformations in Equation (1) are generally rigid (rotations and translations), but their linear combination may not be rigid. As a result, SSD has well-known shortcomings (shown in Figure 2), but despite this, it remains popular for real-time applications due to its simplicity, efficiency, and the established base of modelling tools and rendering systems that support it. Nevertheless, there are many proposed algorithms that address these flaws [Sloan et al. 2001; Mohr and Gleicher 2003a; Kavan et al. 2008; Wang and Phillips 2002; Merry et al. 2006].

2.1.2 Animation space

Animation space is one approach to address the flaws in SSD. In Equation (1), one can define $\mathbf{p}_i = w_i\hat{G}_i^{-1}\hat{\mathbf{v}}$, and thus write SSD as

$$\mathbf{v} = \sum_i G_i\mathbf{p}_i = \begin{pmatrix} G_1 & \cdots & G_b \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_b \end{pmatrix}. \tag{2}$$

The matrix and vector above are labelled G and $\mathbf{p}$, and referred to as the animation projection matrix and the animation-space position of the vertex, respectively. This substitution increases the degrees of freedom, and is not reversible (that is, given an arbitrary $\mathbf{p}_i$, it is not possible to construct corresponding w_i and $\hat{\mathbf{v}}$). The extra degrees of freedom make it possible to overcome the flaws in SSD by judicious selection of the vector $\mathbf{p}$. The only restriction is that $\underline{\mathbf{p}}$, the sum of the homogeneous components of the $\mathbf{p}_i$ vectors, is 1. This is equivalent to the requirement that $\sum w_i = 1$ in SSD, and ensures the convention that $\mathbf{v}$ has a homogeneous weight of 1.

Apart from addressing the flaws in SSD, AS has two advantages that concern us here. Firstly, it is a generalisation of SSD, and thus the algorithms presented in Sections 4 and 5 can be applied to existing models created with SSD. More importantly, those algorithms depend on the $L_{2,2}$ metric of animation space, which measures the root-mean-squared geometric distance between points, with the average taken across all poses. Let $E[f(x)]$ be the expected value of $f(x)$, where x is a random variable. The $L_{2,2}$ distance between $\mathbf{p}$ and $\mathbf{q}$ is defined as a norm on the difference $\mathbf{s} = \mathbf{p} - \mathbf{q}$, with G being the random variable:

$$\begin{aligned} \|\mathbf{s}\|_{2,2} &= \sqrt{E\left[\|G\mathbf{s}\|_2^2\right]} \\ &= \sqrt{E[\mathbf{s}^T G^T G\mathbf{s}]} \\ &= \sqrt{\mathbf{s}^T E[G^T G]\mathbf{s}}. \end{aligned} \tag{3}$$

The expectation $E[G^TG]$ is labelled P, and called the expectation matrix. Merry et al. [2006] show how this matrix may be estimated from a combination of samples (such as from a pre-defined animation) and assumptions about the independence of joints. Note that P depends only on the probability distribution of poses, and is independent of the vertices. Thus, it is practical to estimate P once from thousands or even hundreds of thousands of poses (e.g., every frame in several hours of animation). The number of samples used does not change the cost of evaluating the $L_{2,2}$ metric, which is quadratic in the number of influences on $\mathbf{p}$ and $\mathbf{q}$.

2.2 Level of detail

Most level-of-detail schemes for polygonal models are based on the progressive mesh [Hoppe 1996]. This is a data structure that represents a sequence of simplifications, each of which collapses an edge to a single vertex and removes up to two faces from the model (see Figure 3). A typical progressive mesh scheme uses a priority queue to rank potential edge collapses according to some metric on the local neighbourhood, and repeatedly applies the best collapse [Hoppe 1996; Garland and Heckbert 1997]. This yields a sequence of meshes $M = M_0, M_1, \ldots, M_n$ where each differs from the previous one only in a small neighbourhood. Luebke [2001] provides a survey of the wide variety of level-of-detail schemes, including those based on progressive meshes.

An important consideration in rendering modern characters is texture: a model will typically have high-detail colour information, and sometimes lighting information such as normal or specularity, associated with the surface by means of a parametrisation. If the texture coordinates are poorly approximated, the textures will appear to slide across the surface, even if the geometry is accurate. Appearance-preserving simplification (APS) is a method that addresses this directly: the metric of a simplified representation is a conservative approximation of the maximum distance between the original and simplified representations, with the distances measured between points that have the same texture coordinates [Cohen et al. 1998]. An additional innovation is that the original model may be sampled to produce a normal map, allowing the original normal information to be texture-mapped onto the simplified model and thus yielding high-fidelity lighting. This makes APS advantageous even for models with no existing parametrisation, as by first computing a parametrisation one may take advantage of this high-fidelity lighting.

Sander et al. [2001] make several modifications to APS, including a "memoryless" form of the metric. When considering a candidate collapse that would transform M_i to M_{i+1}, they take the cost of the collapse to be the deviation between M_i and M_{i+1}, rather than the deviation between M_0 and M_{i+1} as in standard APS. This reduces memory requirements and simplifies the implementation. They also use the half-edge collapse, in which one end-point of an edge is collapsed to the other, rather than both end-points being collapsed to a new vertex. This constrains simplifications to lie within the convex hull of the original model, and so convex regions tend to lose volume as they are simplified. The advantage of the half-edge collapse is that there is no need to optimise the location of a newly-introduced vertex, and indeed no new vertices need to be introduced and stored.

3 Related work

While both character animation and LOD are mature fields, there is relatively little research on combining them, and most techniques consider generic animation without taking advantage of the structure of skeletally-animated characters.

Geometry videos [Briceño et al. 2003] extend the idea of geometry images [Gu et al. 2002] to animations. A geometry image is an encoding of a model into an image via a parametrisation: the three colour channels of each pixel contain the X, Y and Z position of a point on the model. A geometry video is simply a sequence of geometry images, which is compressed using techniques from the field of video compression.

Shamir and Pascucci [2001] use a single progressive mesh computed from one pose. They improve on the naïve algorithm by using a directed acyclic graph (DAG) indicating the dependencies between simplifications, and dynamically adjusting a cut through this DAG. This approach has previously been used for view-dependent LOD [Xia and Varshney 1996], but in this case the "view" depends on the view-point in time. Additionally, they propose mechanisms to handle changes in attributes (such as vertex colour), connectivity and topology. Similar approaches have been used by Kircher and Garland [2005] and Payan et al. [2007], which start from a simplification from the first frame and make progressive updates over time.

Deformation-sensitive decimation (DSD) [Mohr and Gleicher 2003b] is closer to our approach. A single progressive mesh is constructed, and only the originally computed representations are used (as opposed to a cut through a dependency graph). Instead of run-time adjustments, the edge collapse metric is modified to take a set of poses into account. The authors use a quadric error metric [Garland and Heckbert 1997], averaged over a number of sample poses. Huang et al. [2006] add an extra term to penalise simplifications of areas that undergo deformation, and also dynamically adjust a graph-cut at run-time to improve the quality of each frame.

The methods discussed above are all designed for a general animation, rather than specifically for a character animation. DeCoro and Rusinkiewicz [2005] adapt DSD to characters animated with SSD, which accommodates two improvements. Firstly, sample poses are automatically generated with stratified random sampling, based on a specified probability distribution for each joint. Secondly, the quadrics for each vertex are transformed back into the reference pose, and may be added to give a single quadric during the initialisation phase. As a result, the more computationally-intensive simplification phase has no additional cost over a static LOD implementation.

Dynamic adjustment of a graph cut is relatively expensive, and is not well-suited to current GPU designs [Dietrich 2000]. The other methods listed here are based on modifications of the error metric, and require no specific run-time manipulation. In other words, the simplifications can be used as replacements for naïve, rest-pose simplifications, with no modifications to the renderer. However, the production of the simplifications during pre-processing is relatively expensive due to the sampling, as some part of the algorithm is at least $O(SV)$ for S samples and V vertices. Although DeCoro and Rusinkiewicz [2005] report reasonable results with 16 samples, it would be more satisfying to separate the estimation of the pose-space probability distribution from per-vertex/edge operations. Animation space achieves this: only certain moments of the distribution are needed to compute the $L_{2,2}$ metric, and these can be computed (analytically or by numerical means) from a specified distribution, or estimated from a supplied animation sequence, where it is practical to use thousands of samples.

If sampling is used, then our algorithm has complexity $O(SB + B^2 + I^2V + V \log V)$ for S pose samples, B bones, I influences per vertex and V vertices; the terms are respectively for estimating per-bone probability distributions, computing P from these, measurements of the $L_{2,2}$ metric, and for maintaining the priority queue. In a typical model, B is quite small (under 100) and I is even smaller (under 10) regardless of the number of vertices, so in the case of many samples and vertices, we expect our algorithm to perform significantly better than one with an $O(SV)$ term.

4 Progressive meshes in animation space

We base our approach on appearance-preserving simplification (APS). Apart from its intrinsic advantages, APS works well in this context because it uses a parametrisation to determine the correspondence between different representations of a model. This would otherwise be quite difficult and costly to determine in a high-dimensional space such as animation space.

Figure 3 shows the neighbourhood of an edge collapse in parameter space. Within each region, or cell, of the diagram, both M_i

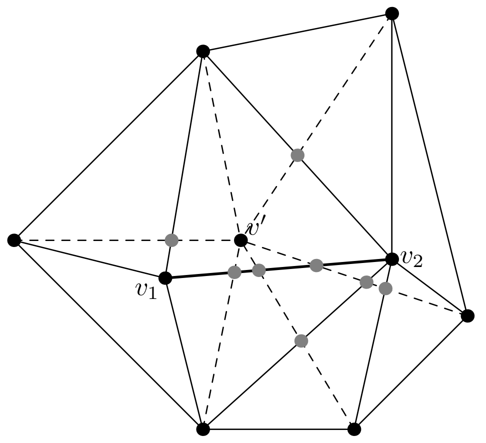


Figure 3: *Appearance-preserving simplification, showing an edge collapse $v_1v_2 \rightarrow v'$ in parametric space. Dashed lines represent the new neighbourhood, and dots represent the corners of cells.*

(the mesh immediately prior to the collapse) and M_{i+1} (the mesh produced from M_i by the collapse) are linear in the parametric coordinates, and hence so is the vector representing the difference between corresponding points. Since these cells are convex, it follows that the maximum difference between M_i and M_{i+1} occurs at one of the cell corners, marked by dots in the figure.

This is all that is needed to compute the memoryless APS metric for a candidate collapse. While Cohen et al. [1998] try several possibilities for placing v', Sander et al. [2001] restrict v' to coincide with either v_1 or v_2, and we have followed this as it reduces the number of cell corners to consider, as well as simplifying the implementation.

In animation space, the situation is no different: we simply replace the Euclidean metric with the $L_{2,2}$ metric when measuring the deviation at cell corners.

The standard form of APS, in which the deviation is measured between M_0 and M_{i+1}, is more complicated. Measuring the exact Hausdorff distance is prohibitively expensive, so a conservative approximation is used instead. A bounding box is associated with each face, which bounds the vector between any point on that face and the corresponding point in M_0. These are combined with the offset vectors at the cell corners to produce updated bounding boxes for the faces in the new neighbourhood. The primary advantage of this form is that it provides guaranteed error bounds. However, the memoryless form generally produces results of similar quality, while being simpler to implement and more time- and memory-efficient to execute [Sander et al. 2001].

The standard form of APS can be adapted to animation space, but doing so accentuates the disadvantages of this form. Bounding boxes must now be stored and manipulated in animation space, which has four dimensions per bone. The approximation quality is also poor if the bounding boxes are axis-aligned, because the axes are not conjugate with respect to the expectation matrix P (recall that the $L_{2,2}$ norm is defined as $\|\mathbf{s}\|_{2,2}^2 = \mathbf{s}^T P\mathbf{s}$). Instead, we use bounding parallelepipeds with sides parallel to the eigenvectors of P, which produces better results. Unfortunately, it destroys the sparsity of animation-space coordinates, further increasing time and memory requirements.

5 Influence simplification

In adapting APS to animation space, we have thus far treated vertices as atomic entities which all cost the same to store or render. In animation space, however, the cost to store a vertex is directly proportional to the number of bones that influence it. The rendering cost is more implementation-dependent, but the number of influences is still a significant consideration.

We now describe *influence simplification*, a method for reducing the number of influences on a vertex in an optimal way. Given a vertex, an influence simplification removes one influence from the vertex, and also modifies the remaining influences so that the new vertex is as close to the original as possible in the $L_{2,2}$ metric.

Let $\mathbf{p}$ be a vertex in animation space. Without loss of generality, suppose $\mathbf{p}$ is influenced by bones $1, 2, \dots, k$, and that we wish to eliminate the influence from bone k (we try to eliminate each influence of $\mathbf{p}$ in turn, and take the simplification with the minimum deviation). The simplification will replace $\mathbf{p}$ with $\mathbf{p} + \mathbf{s}$, subject to the constraints:

1. $\mathbf{s}$ is influenced by (at most) bones 1 through k.
2. $\mathbf{s}_k = -\mathbf{p}_k$, to cancel the influence on $\mathbf{p}$.
3. $\mathbf{s}$ is a vector, i.e., $\underline{\mathbf{s}} = 0$ (recall that $\underline{\mathbf{s}}$ is the sum of the homogeneous components of $\mathbf{s}$).
4. $\|\mathbf{s}\|_{2,2}$ is minimal subject to the above.

Both $\mathbf{p}$ and $\mathbf{s}$ have non-zero coordinates only in those dimensions of animation space corresponding to the first k bones, and working only within this subspace improves the performance of the optimisation. Let $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{s}}$ be the coordinates of $\mathbf{p}$ and $\mathbf{s}$ in this subspace, and $\tilde{P}$ be the submatrix of P consisting of the first k rows and columns. The optimisation can then be reformulated as minimising $\tilde{\mathbf{s}}^T\tilde{P}\tilde{\mathbf{s}}$ subject to $A\tilde{\mathbf{s}} = \mathbf{b}$, where A is a $5 \times 4k$ matrix and b a 5-vector that together encode constraints 2 and 3 above.

This is a relatively straightforward optimisation problem when $\tilde{P}$ is non-singular. Write $\tilde{P}$ as UDU^T, where U is orthogonal and D is diagonal. This is possible because $\tilde{\mathbf{s}}^T\tilde{P}\tilde{\mathbf{s}} \geq 0$ for all $\tilde{\mathbf{s}}$, and hence $\tilde{P}$ is non-negative definite symmetric. Let $C = \tilde{P}^{\frac{1}{2}} = UD^{\frac{1}{2}}U^T$, let $\tilde{\mathbf{s}}' = C\tilde{\mathbf{s}}$ and let $A' = AC^{-1}$. The problem may now be rewritten as minimising $\|\tilde{\mathbf{s}}'\|$ (the Euclidean norm) subject to $A'\tilde{\mathbf{s}}' = \mathbf{b}$, which is simply a matter of projecting the origin onto the subspace defined by the constraint. The same approach applies when it is singular (which is not unlikely [Merry et al. 2006]), but some extra steps must be taken to regularise the problem.

This process is quite slow. The computational cost can be amortised by noting that the most expensive steps, namely computing the diagonalisation of $\tilde{P}$ and manipulating A, do not depend on $\mathbf{p}$ but only on the set of bones that influence it and the influence that is to be removed. We cache the results of these expensive computations and re-use them on nearby vertices, which often have the same influences. Cache hit rates are very close to 100%, and this optimisation produces at least an order-of-magnitude speedup.

5.1 Combination with progressive meshes

Apart from defining the optimisation, the $L_{2,2}$ metric can be used to measure the deviation of an influence simplification. Since both APS and influence simplification define deviations in terms of the $L_{2,2}$ metric between two points in correspondence, the two types

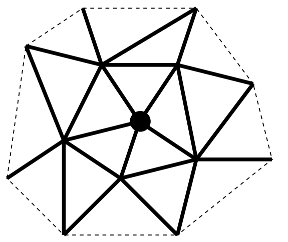


Figure 4: *Edge updates from an influence simplification. When the central vertex is simplified, the edge collapses corresponding to the solid edges must be updated.*

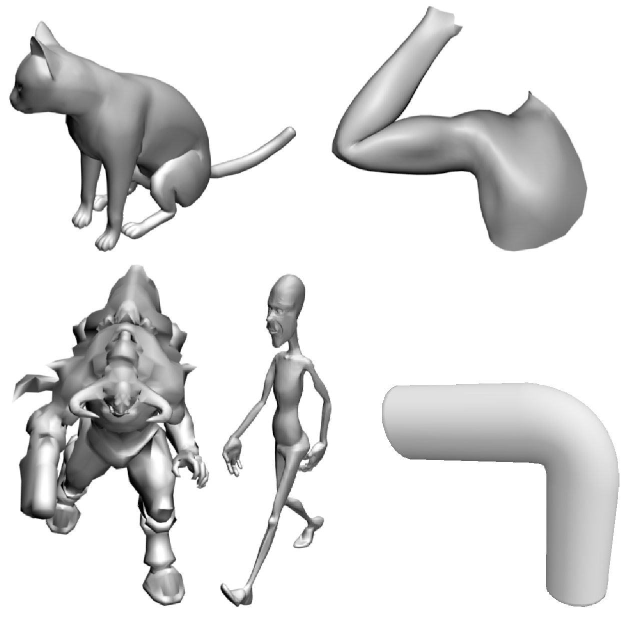


Figure 5: *Test models: cat, arm, cyberdemon, mancandy and cylinder. The horse model is shown in Figure 1.*

of simplification can be interleaved in a single sequence of simplifications. For memoryless APS, this is quite straightforward. The priority queue of potential simplifications now contains both edge collapses and influence collapses. When a half-edge collapse is performed, the vertex that was eliminated must also be removed from the priority queue, apart from the usual book-keeping common to all progressive mesh algorithms. When an influence is removed from a vertex, every edge incident on a triangle containing this vertex must be updated in the priority queue (see Figure 4), and if the vertex still has more than one influence, it must be re-evaluated for further simplification.

Similarly, it is possible to combine influence simplification with standard APS, yielding collapse costs that are a conservative bound on the $L_{2,2}$ distance between any two corresponding points, by making appropriate updates to the per-face bounding boxes. However, we will see that this produces poor results, so we will not elaborate further on it.

6 Results

6.1 Quality

We used six models to test our implementation, shown in Figures 1 and 5. Cylinder is an artificial example (a simple cylinder with a bend in the middle) produced by fitting an animation-space model to a set of examples [Merry et al. 2006]; horse, cat and arm are more realistic models produced with the same fitting process. Cyberdemon is a character from Doom III (copyright Id software and used with permission), and mancandy is a demonstration model from the Blender test suite, with two steps of Catmull-Clark subdivision applied.

In Figure 1, we compare our methods of simplification against traditional LOD based only on the rest pose. In all cases, we are using memoryless APS, and we have chosen representations whose number of influences are as similar as possible. It is clear that in this (non-rest) pose, animation-space simplification correctly preserves detail in the joints, and that influence simplification allows more geometric detail to be kept.

In 1(c) we also show a worst-case scenario for influence simplification: a vertex in the chest is approximated as well as possible, but in this pose the error is relatively large, and leads to a protrusion. Of course, this level of detail would normally only be used for distant models, and so this error would not be noticeable.

The rest pose of the mancandy model has most joints in a bent pose, so using the $L_{2,2}$ metric contributes little. Figure 6 shows a close-up of the hand, where this metric improves the shape of the wrist and of the bottom finger. Influence simplification significantly improves the curvature of the palm.

Figure 7 shows graphs of the root-mean-square (RMS) geometric deviation of a number of models simplified using three methods, plotted against the number of influences in the model. The means are taken over both the surface of the model and the frames of an animation. Because the errors span many orders of magnitude, the errors are shown relative to rest-pose simplification for clarity.

The horse and cat models indicate that influence simplification combined with standard APS (dashed red) does not give reliable quality, although the results are better for the other models. The solid lines represent memoryless APS, which is the preferred form for our method. Simplification in animation space, even without influence simplification, consistently produces results that are better overall, and are slightly worse only with extreme simplification of some of the models.

Introducing influence simplification has more variable results. Generally it makes an improvement, in some cases by an order of magnitude, but in the case of mancandy the results are worse for intermediate levels of detail. We conjecture that the disappointing results for mancandy are caused by large translation components in the model's joint matrices, which are poorly handled in our estimation of P from sample poses. Since large translations are anatomically implausible, we do not expect this to be a serious problem in general. It is also worth noting that for the influence counts in question, the absolute error is less than 10^{-4} — less than the quantisation error of many "lossless" compression schemes [Alliez and Gotsman 2004] — and so the difference is unlikely to be visible even when the model is in the foreground.

Influence simplification is particularly beneficial in two areas. During the initial simplification steps (to the right in each graph), superfluous influences may be eliminated at very little cost. While this effect is most obvious in the artificial cylinder model, it is also

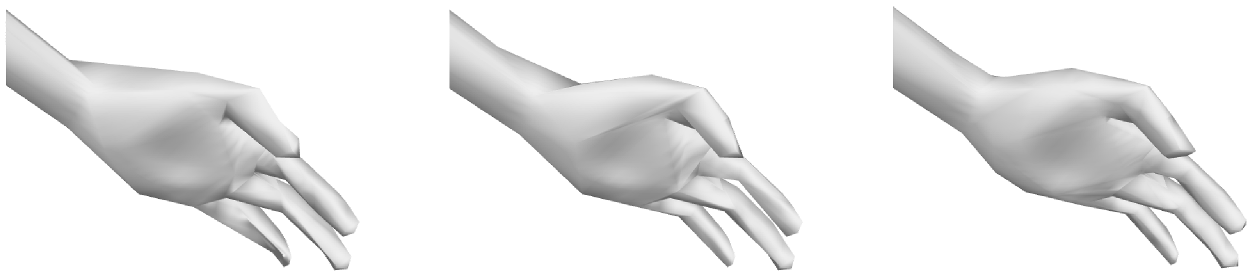

(a) rest-pose LOD, 3000 influences (b) AS LOD, 2997 influences (c) inf. simplification, 3000 influences

Figure 6: *Mancandy's hand at a low level of detail. The most visible artefacts are in the silhouettes of the upper wrist and lower palm.*

clearly present in the real-world cyberdemon model, where almost 250 influences are removed before the graph is even visible above the axis. This also suggests that influence simplification may be used as a method to restrict the number of influences in a model, possibly allowing different levels of detail to be selected depending on hardware capabilities.

The second area in which influence simplification excels is extreme simplification. For an influence count where a pure progressive-mesh algorithm is forced to apply all legal edge collapses, using influence simplification allows some detail to be preserved while instead eliminating some influences. Furthermore, influence simplification is able to produce representations with fewer total influences than is possible with only edge collapses. Since APS cannot apply collapses that alter the texture space covered by the model, this is not a trivial concern.

6.2 User tests

Geometric error is not necessarily a good indicator of perceived quality. For example, displacing every vertex by a fixed amount will yield a large geometric error but there will be no apparent loss of quality, while displacing only a random 50% of the vertices by the same offset will yield less geometric error but will look far worse. We conducted tests with users to validate the results of the geometric error tests.

Each test sample involved showing a user a pair of short video clips (7–8 seconds). Each video showed a number of copies (typically 100) of the same animated character. Within each pair, the content was the same, as was the camera path (an inward spiral to show a range of angles and distances), and only the LOD mechanism differed. Rather than showing the videos side-by-side, which would have allowed the user to make pixel-level comparisons, we showed the videos sequentially so that they would be evaluated qualitatively. After the videos were shown, the user was asked to select which one was better, with a forced choice.

For each of the test models except the cylinder, we created three progressive meshes using

(a) edge collapses, ranked on the rest pose;

(b) edge collapses, ranked by the $L_{2,2}$ metric;

(c) edge collapses and influence simplifications, ranked by the $L_{2,2}$ metric.

In every case, we used memoryless error metrics. In the first set of tests, we compared (a) to (b), with an error tolerance of 1.5, 3 or 6 pixels for (b), and a tolerance for (a) that yielded, as closely as possible, the same total number of influences across all frames of the video. The second set compared (b) to (c) similarly, although some samples had to be dropped as without influence simplification, it was impossible to reduce (b) to an equivalent influence count to (c).

We had 23 volunteer test subjects, mostly students and staff at a university. For each combination of subject, model, test type and pixel tolerance we took two samples, one with the order of the two videos reversed to control for ordering bias. For each subject, the order of the samples was randomised, and also interleaved with samples from a third test type which is not the subject of this paper. In a pilot study, we found strong evidence of a learning effect, where users would learn to identify specific artefacts, and were less likely to make random choices in later samples. To control for this, we first conducted a *training* phase, consisting of a third sample of each type (with the order of the two videos being random). These samples were collected exactly as for the experiment phase and users were not made aware of any difference, but the results from the training phase were discarded.

In the first experiment (rest-pose versus AS), we expected that the error tolerance would be an influencing factor, with more statistical significance at higher error tolerances (since with a zero or small tolerance, the result is indistinguishable from the ideal, irrespective of the LOD algorithm). We were surprised to find that the reverse was true. At 1.5 pixels, there was a significant preference ($p < 0.002$) for animation-space simplification over rest-pose simplification, while at 3 and 6 pixels there was no statistically significant result. We conjecture that at 1.5 pixels the errors in the rest-pose simplifications are just noticeable to users but those of animation-space simplifications are not, while at higher tolerances, users are unable to distinguish between large errors with little relative difference.

In the second experiment (with influence simplification versus without), the choice of scene was highly significant. This is not surprising, as some models have more redundant influences than others. Only the horse and mancandy models showed statistically significant results, with users preferring influence simplification. This is in spite of the apparently poor performance of influence simplification on mancandy shown in Figure 7, confirming that the central areas of this graph (where influence simplification performs poorly, but where absolute errors are minuscule) are less important.

6.3 Off-line performance

Table 1 summarises properties of the models, and shows the impact of our improvements on the time required to produce progressive meshes. Although the animation-space metric is more expensive to compute, a substantial portion of the running time is devoted to other calculations such as identifying the cell corners and maintain-

Table 1: *Relative pre-processing performance of the memoryless simplification methods. 3D is traditional simplification of the rest pose, ignoring animation; AS is animation-space edge collapses; and AS+IS is AS with the addition of influence simplifications. Cache miss rate is the number of times that matrix factorisations are not found in the cache during influence simplification.*

Model	Vertices	Bones	Influences	Time (s)			Ratios		Cache
			per bone	3D	AS	AS+IS	AS/3D	AS+IS/3D	miss rate
arm	1600	4	2.1	1.7	2.2	4.3	1.3	2.5	0.0016%
cyberdemon	2282	65	1.5	1.6	2.1	4.8	1.3	3.1	0.26%
cylinder	2426	2	2.0	2.6	3.0	5.9	1.2	2.3	0.00044%
cat	7207	26	2.2	7.8	10.3	18.8	1.3	2.4	0.0099%
horse	8431	25	2.6	9.5	13.7	26.3	1.4	2.8	0.018%
mancandy	42654	96	2.4	64.6	92.6	203.8	1.4	3.2	0.038%

ing the priority queue, and thus using the AS metric adds at most 50% overhead. It should be noted that in all three methods we used the same code, whereas an implementation designed specifically for the 3D case may be expected to perform slightly better.

Influence simplification is more expensive, as we must solve an optimisation problem for each potential simplification. We cache factorisations of $\tilde{P}$ and A, for the last 1024 influence sets encountered. The low cache miss rates shown in Table 1 are instrumental in maintaining performance, and as a result, running times increase by a factor of at most 3.2. We found that a larger cache has diminishing returns, and the cost of searching the cache becomes a disadvantage.

6.4 Rendering performance

Influence simplification somewhat complicates rendering, since the resulting models are not representable within the SSD framework. We implemented a renderer to measure the impact on performance. It uses geomorphing [Hoppe 1996] to smoothly interpolate between levels of detail, and tangent-space normal maps [Peercy et al. 1997] to reconstruct the original lighting. This requires six 4-vectors to be used in vertex transformation for each influence on a bone: a position and two tangents for each of two levels of detail. With the 16 per-vertex attributes guaranteed by OpenGL, this would limit a straightforward implementation to two influences per vertex. To avoid this limit, we encode the animation-space positions and tangents into textures, which we access from the vertex shader.

Merry et al. [2006] have previously shown that using animation-space coordinates for rendering actually improves performance over a straightforward SSD renderer, as the weight is pre-multiplied. Table 2 shows the rendering performance with the method described above at 1280×1024 on a GeForce 8800GTX and a Core2Duo E6600 (clock speed 2.4GHz). With the exception of the 1020-instance horse scene, the scenes and camera paths are the same as those used in user testing.

The apparent drop in performance when using animation-space simplification is due to the way error is measured: when rendering the naïve simplifications, the tolerance is for a 2-pixel error in the rest pose, but the actual error may be much greater, whereas the animation-space levels of detail are chosen for an average error of 2 pixels across all poses. Thus, while sometimes slightly slower, these representations will more accurately meet the nominal pixel tolerance.

It should be noted that the same code-paths are used when LOD is disabled, and in particular, geomorphing is still done. It is thus likely the speedups will not be quite as dramatic in practise.

We had hoped that influence simplification would improve rendering performance, due to fewer influences needed for a given tolerance, and also because further simplification is possible. While there are indeed fewer influences rendered, Table 2 suggests that this is not the bottleneck in the rendering pipeline. Our implementation has a relatively high per-object overhead (around 15μs), but recent extensions to the OpenGL API, such as instancing [Gold 2006] and bindable uniforms [Brown and Lichtenbelt 2008] have the potential to reduce the overheads involved and may make influence simplification more beneficial.

Table 2: *Rendering performance in frames per second, for a nominal 2-pixel error tolerance. Captions are as for Table 1.*

Model	Copies	None	3D	AS	AS+IS
arm	100	321	720	698	698
cyberdemon	102	146	215	212	214
cylinder	100	329	963	974	976
cat	100	55	311	296	296
horse	100	32	232	214	211
	1020	3	25	25	25
mancandy	100	6	40	40	40

7 Conclusions

We have demonstrated two improvements to the quality of LOD for articulated characters. The first is to perform computations in animation space. Unlike previous approaches to the problem, no sampling of the pose space is required. Sampling may be used to estimate the matrix P, but this is done once rather than per-vertex, and so it is practical to use very large sample sizes. While animation-space computations allow this method to be applied to animation-space models, it may also be applied to SSD models, and the output is again a sequence of SSD models due to the half-edge collapse. Thus, while our method is limited to articulated models, it is not necessary to use animation space in either modelling or rendering. As with previous work that samples the pose space [Mohr and Gleicher 2003b; DeCoro and Rusinkiewicz 2005], the algorithm is applied entirely off-line, and there is no special work to be done during rendering. We have not done a direct comparison to this prior work, but we expect visual quality to be at least as good due to the analytic error computations and the use of appearance-preserving simplification.

The second contribution is influence simplification, which eliminates influences rather than geometry. This improves quality and allows for more extreme simplification than is otherwise possible. It is also useful for eliminating redundant influences in a base model (independently of any run-time LOD), e.g., to meet a hardware limitation on the number of influences per vertex. The disadvantages are the additional computational cost (but only during preprocessing), and the fact that output models may not be representable within SSD. This makes it undesirable for compression or progressive transmission of SSD models, but it is useful for rendering due to the improved quality [Merry et al. 2006].

Acknowledgements

Without test models, it would have been impossible to evaluate this research. We are indebted to Robert Sumner, J. P. Lewis, Id Software and the Blender Foundation for making their data available.

References

ALLIEZ, P., AND GOTSMAN, C. 2004. Recent advances in compression of 3D meshes. In *Advances in Multiresolution for Geometric Modelling*. Springer.

BRICEÑO, H. M., SANDER, P. V., MCMILLAN, L., GORTLER, S., AND HOPPE, H. 2003. Geometry videos: a new representation for 3D animations. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 136–146.

BROWN, P., AND LICHTENBELT, B. 2008. *GL_EXT_bindable_uniform extension*. `http://www.opengl.org/registry/specs/EXT/bindable_uniform.txt`.

COHEN, J., OLANO, M., AND MANOCHA, D. 1998. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, 115–122.

COLLINS, G., AND HILTON, A. 2001. Modelling for character animation. *Software Focus 2*, 2, 44–51.

DECORO, C., AND RUSINKIEWICZ, S. 2005. Pose-independent simplification of articulated meshes. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 17–24.

DIETRICH, S., 2000. Optimizing for hardware transform and lighting.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 209–216.

GOLD, M. 2006. *GL_EXT_draw_instanced extension*. `http://www.opengl.org/registry/specs/EXT/draw_instanced.txt`.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 355–361.

HOPPE, H. 1996. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, 99–108.

HUANG, F.-C., CHEN, B.-Y., AND CHUANG, Y.-Y. 2006. Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 53–62.

KAVAN, L., COLLINS, S., ŽÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. ACM Press, New York, NY, USA, vol. 27.

KIRCHER, S., AND GARLAND, M. 2005. Progressive multiresolution meshes for deforming surfaces. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, 191–200.

LUEBKE, D. P. 2001. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl. 21*, 3, 24–35.

MERRY, B., MARAIS, P., AND GAIN, J. 2006. Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4, 1400–1423.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graphics 22*, 3, 562–568.

MOHR, A., AND GLEICHER, M. 2003. Deformation sensitive decimation. Tech. Rep. 4/7/2003, University of Wisconsin, Madison.

PAYAN, F., HAHMANN, S., AND BONNEAU, G.-P. 2007. Deforming surface simplification based on dynamic geometry sampling. In *SMI '07: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2007*, IEEE Computer Society, Washington, DC, USA, 71–80.

PEERCY, M., AIREY, J., AND CABRAL, B. 1997. Efficient bump mapping hardware. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 303–306.

SANDER, P. V., SNYDER, J., GORTLER, S. J., AND HOPPE, H. 2001. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, 409–416.

SHAMIR, A., AND PASCUCCI, V. 2001. Temporal and spatial level of details for dynamic meshes. In *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, 77–84.

SLOAN, P.-P. J., ROSE, III, C. F., AND COHEN, M. F. 2001. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, 135–143.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM Press, 129–138.

XIA, J. C., AND VARSHNEY, A. 1996. Dynamic view-dependent simplification for polygonal models. In *Proceedings of the 7th conference on Visualization '96*, IEEE Computer Society Press, 327–ff.

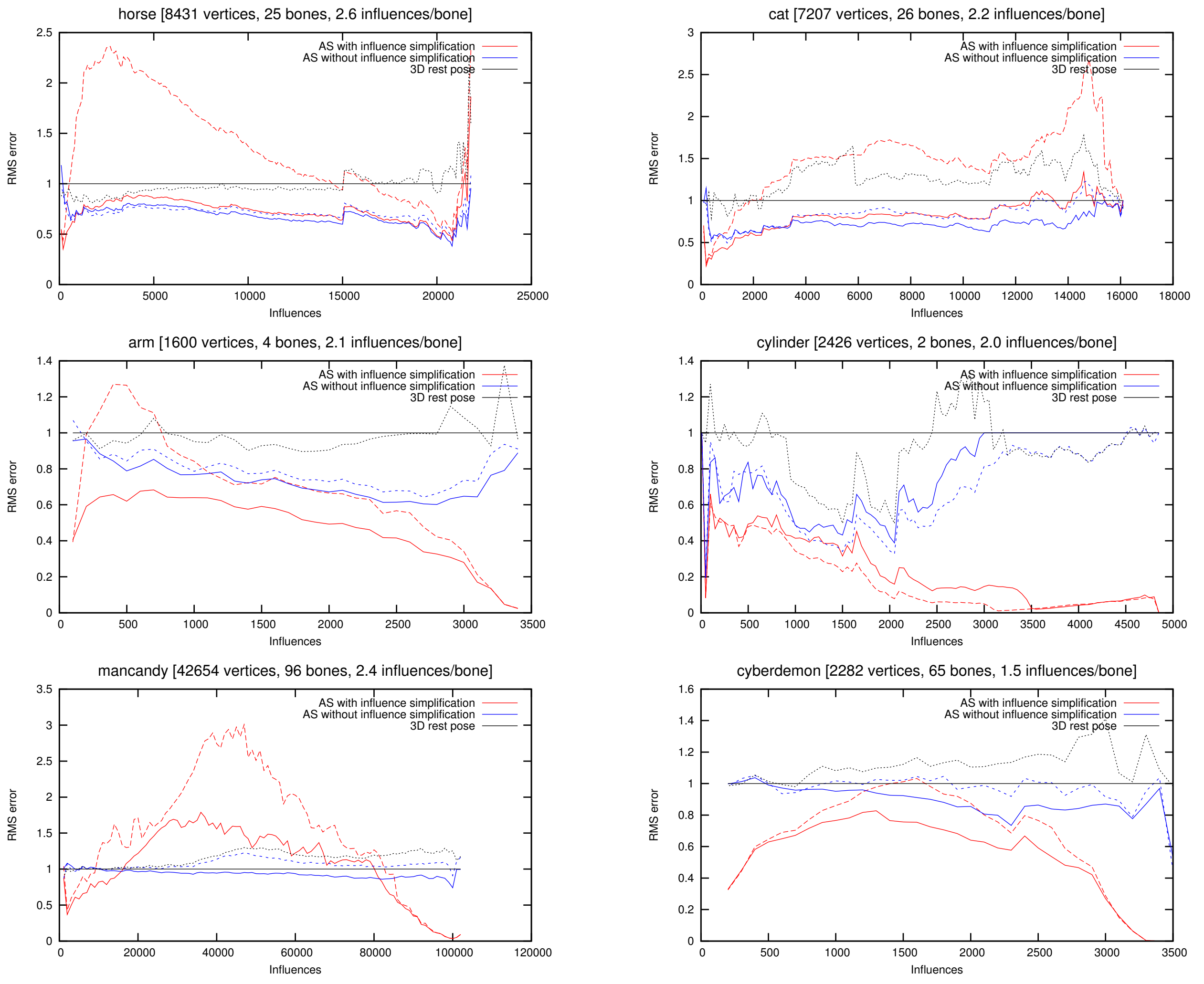


Figure 7: *Simplification errors, normalised relative to those for 3D rest-pose simplification, for the six models. The dashed lines show standard APS, while the solid lines of corresponding colours show memoryless APS. Note that simplification proceeds right-to-left along each graph.*