Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

A Rigid Transform Basis for Animation Compression and Level of Detail

G. Collins and A. Hilton

Centre for Vision, Speech and Signal Processing
University of Surrey, Guildford GU25XH, UK
a.hilton@surrey.ac.uk http://www.ee.surrey.ac.uk/Research/VSSP/3DVision

Abstract

We present a scheme for achieving level of detail and compression for animation sequences with known constant connectivity. We suggest compression is useful to automatically create low levels of detail in animations which may be more compressed than the original animation parameters and for high levels of detail where the original animation is expensive to compute. Our scheme is based on spatial segmentation of a base mesh into rigidly transforming segments and then temporal aggregation of these transformations. The result will approximate the given animation within a user specified tolerance which can be adjusted to give the required level of detail. A spatio-temporal smoothing algorithm is used on decoding to give acceptable animations. We show that the rigid transformation basis will span the space of all animations. We also show that the algorithm will converge to the specified tolerance. The algorithm is applied to several examples of synthetic animation and rate distortion curves are given which show that in some cases, the scheme outperforms current compressors.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

We consider the representation of synthetic animations consisting of sequences of meshes with constant connectivity across time. Our goal is to provide automatic level of detail and compression by using a basis of rigidly transforming subsets of the mesh.

For such animations an effective compression is already available in the form of the objects used and operations performed by the animator. Most animation packages store their data parametrically so that, for example, to encode a bouncing ball animation only a few parameters (ball diameter, elasticity, gravitational constant) are needed. We refer to this as the animator's parameterisation. To compress this by considering a sequence of meshes would seem perverse since it would seem that the most efficient compression will be given by the few parameters needed to describe the physics and geometry of the situation. However, there are three cases where the animator's parameterisation is not more efficient and another basis for animation is needed. First, for computationally expensive animations, such as physics based simulations, the parameters of the simulation will give a more compact representation but the decoding (solving of the simulation) will take a prohibitively long time. For analytic (captured) animations there is no ready animator's parameterisation and so a representation (which ought to be more efficient than a sequence of meshes) must be formed just to display the animation. In this case there is no temporal correspondence between vertices and we do not consider this here although our work may be extended to solve the correspondence problem also. Lastly, automatic level of detail is rarely possible with the animator's parameterisation since there is no automatic way to decide which parameters are the most significant.

With these points in mind, we concentrate on the problem of providing a controllable level of detail parameterisation of synthetic animations using a simple rigid transform (RT) basis for animation. We seek to approximate an animation with a sequence of rigid transformations applied to regions of a base mesh. We note that we can always find such a basis which will span a given animation. This is because we can always segment the animation into pieces of single vertices and code the RTs as translations of each vertex from

frame to frame (without compression). We approximate the animation by segmenting it into larger pieces which are then rigidly transformed. To optimise the compression, we wish to have as many large pieces as possible while maintaining an error within a user specified tolerance. Further compression is achieved by then aggregating rigid transformations between time steps where there is little movement of the segment. The resulting reconstructions, although demonstrably accurate in the RMSE sense, may not be smooth in space or time. We therefore present a spatio-temporal smoothing which can be applied on reconstruction. We give examples to demonstrate progressive levels of detail of animation and give rate distortion curves of the compression which show that the basis is competitive with other schemes. We also demonstrate automatic level of detail which, for low levels of detail, is compressed more efficiently than the animator's parameterisation.

Our contribution to the field is

- Coding of animations with a simple RT basis which spans a given animation without residual.
- Weighted least squares segmenting of a base mesh into rigidly transformable pieces.
- Exploiting temporal coherence in the animation by aggregating transformations which are similar over time.
- Spatio-temporal smoothing
- Rate distortion analysis

2. Previous Research

Both the graphics and the vision communities have contributed to the background for the relatively new field of the analysis of dynamic meshes. From graphics the need for level of detail, re-meshing and compression of static meshes has given rise to a field of research into mesh parameterisation [10, 7, 9]. The vision community have for a long time been dealing with compression in video by tracking rigid and non-rigid scene features [4] and by principal component analysis [3].

Due to a lack of captured dynamic 3D data, research into dynamic meshes has concentrated on synthetic animations with constant topology and hence correspondence issues have been by-passed. As a result the full power of these graphics and vision algorithms - optical flow, re-meshing, tracking - has not yet been applied to the 3D case. There has, however, been a research effort which has been motivated by the need for LOD and compression algorithms to be used in the dynamic setting. Guenter et al [6], for example, have analysed the 3D positions of face markers to obtain a compressed representation.

Lengyel [11] has introduced the subject of time-dependent geometry compression. He raises the question of what basis should be chosen to represent animation and considers a number of different candidates for bases: free-form deformations, key-shapes, weighted trajectories and skinning before describing an algorithm which codes the animation as a set of dynamic affine transformations on different sections of a base mesh (although he suggests that an improved scheme would use a hybrid of bases). After these affinely transforming segments have been decomposed out of the animation, the residuals of the motion are coded. He goes on to discuss a sophisticated quantisation of both the residuals and the affine transformations.

Purely geometric basis have been considered both by Alexa and Muller [1] and by Briceño et al. [2]. Alexa and Muller affinely transform each animation frame to the position of the first frame and then use PCA on the geometry of each frame to obtain a basis consisting of meshes. However, apart from the transformation to the centre of mass, their basis is a static geometry entirely decoupled from the dynamic mesh. While this may be a highly robust method it cannot claim to be an optimal basis for the *time*- dependent geometry. The result is that a lot of possibly correlated mesh geometries are stored. Brisceno et al. [2] also encode each mesh separately as a geometry image [5]. The result can be compressed using standard video compression. Again any temporal coherence that can be modelled as a basic transformation will not be exploited.

Shamir and Pascucci's [12] scheme exploits both temporal and spatial coherence. They take Lengyel's scheme and embed it in a multiresolution data structure (TDAG) which can be accessed temporally or spatially allowing the resolution of each frame to change as a standard static level of detail scheme would. The change in resolution of each frame is driven the cost of performing a decimation in the mesh and the improvement in the mesh. Such a data structure would address the coupling of the spatial and temporal components of any scheme but still leaves open the question of what basis to choose.

In choosing the most efficient basis there is a trade-off between the size of the base vectors (for example 12 parameters for affine transformations) and the dimension of the vector space (which will be larger for less sophisticated transformations). In this paper we go further than Lengyel's affine transformations in noticing that any animation can be coded as a set of rigid transformations (and hence with only 6 parameters) on a segmented base mesh provided the segments are small enough (and hence the dimension of the space larger) In the most trivial lossless case, each vertex could be translated onto the vertex in the next frame (although, in this case, no temporal coherence would be exploited) At the other extreme (as in Alexa and Muller [1]) an entire animated object could be transformed to a central position giving a very coarse approximation. In between these extremes there is scope for a level of detail approach.

3. Preliminaries

At frame i the coordinates of a mesh with N vertices are given by the homogeneous coordinates:

$$V^i = \{\mathbf{v}_j^i\} \qquad \text{for } j = 1 \ldots N \text{ and for } i = 1 \ldots M.$$

where N is the number of vertices of the mesh and M is the number of frames in the sequence. Here, spatial dependence is given by the subscript j and time dependence is denoted by the superscript i.

We wish to represent the animation by a base mesh (typically the first mesh of the sequence) $B = \{\mathbf{b}_j\}$ for $j = 1 \ldots N$ with the same connectivity as each V^i and a series of $M \times N$ rigid transformations (RTs) given by 4×4 matrices T_j^i composed of rotations R_j^i and translations $\mathbf{t}_j^i$. The approximated vertex of the animation at time frame i is then

$$\mathbf{u}_j^i = T_j^i \mathbf{b}_j$$

We ensure that each base vertex $\mathbf{b}_j$ has exactly one RT, T_j^i at each frame i. A lossless compression is possible since rigid transformations will span the animation. This is apparent since there exists a base mesh and a series of RTs which will (without compression) give the animation exactly, namely

$$\begin{aligned} \mathbf{b}_j &= \mathbf{v}_j^0 \\ R_j^i &= I \\ \mathbf{t}_j^i = \delta\mathbf{v}_j^i &= \mathbf{v}_j^i - \mathbf{v}_j^{i-1}. \\ \delta\mathbf{v}_j^0 &= 0 \end{aligned} \tag{1}$$

This representation, as it stands, represents no compression over the original data. Compression is achieved by segmenting the series of RTs into a $m \times n$ series of distinct RTs where $m < M$ and $n < N$. The mesh B is first spatially segmented into distinct pieces B_l so that $B = \{B_l, l = 1 \ldots n\}$ and a set of $M \times n$ rigid transforms. We then segment temporally by aggregating the M transforms into m transforms to give the $m \times n$ compression.

4. Error Metrics

For a level of detail formulation we require the approximation be bounded by a given tolerance tol in the sense of some error metric. We compare two error metrics with the usual proviso that the metrics are purely geometric and are therefore not necessarily the metric to give the "best" visual results. An alternative may be to consider a metric which takes into account appearance attributes such as that employed by Hoppe [8]. We use a Euclidean metric which averages error in the Euclidean norm over time and also a maximum metric which is the largest error over time and space so that

$$\|e\| = \|e\|_2 = \frac{1}{M}\sum_{i=1}^{M}\sqrt{\frac{1}{N}\sum_{j=1}^{N}|T_j^i\mathbf{b}_j - \mathbf{v}_j^i|^2} \tag{2}$$

$$\text{or } \|e\|_{max} = \max_i\{\max_j\{|T_j^i\mathbf{b}_j - \mathbf{v}_j^i|\}\} < tol \tag{3}$$

where tol is the user specified tolerance which defines the level of detail of the approximation.

5. Spatial Segmentation

We now describe an algorithm to segment the base mesh into pieces with different series of RTs applied to them. Our problem is to find the minimum number of RTs in such a way that the error is still bounded by the tolerance tol.

For each time frame i we solve a binary weighted least squares problem to solve for an average transform T_j^i and the vertices which it applies to. The weights allow us to determine which vertex belongs to the piece under consideration. If the error at a vertex is larger than tol then that vertex should not belong to the piece and its weight is set to zero and so does not contribute in the next iteration of the minimisation. Otherwise it is set to 1. The minimisation is then performed again to calculate a new RT with the new weights and this process is iterated until convergence is reached.

Specifically, to define a series of RTs for a new piece B_l, we start at frame 0 with those vertices which have not yet been segmented:

$$B_l^0 = \{B - B_s \text{ for } s = 0 \ldots l-1\}.$$

Proceeding through each frame i the piece B_l^i of size N' will have vertices removed until the end of the animation is reached and so $B_l^M = B_l$.

At each time frame i, we compute a single transformation T^k and a weight $w^k(j)$ where k is the iteration number and j refers to vertex $\mathbf{b}_j$. We wish to find T^k and $w^k(j)$ such that the error

$$e^k = \sqrt{\sum_{j=1}^{N'} w^k(j)|(T^k\mathbf{b}_j - \mathbf{v}_j^i)|^2 / \sum_{j=1}^{N'} w^k(j)} < tol \qquad \text{for } j = 1 \ldots N' \tag{4}$$

is minimised. The weights will determine which vertices will be included in the current piece. In order to obtain the least possible number of pieces and hence the optimal compression, we wish to have the largest value of $\|w\|_2$ for which the least squares problem will converge to tol. Each weight is set initially to 1 and the minimisation is performed using SVD to find $\min_{T^k}\{e^k\}$ and T^k. A weight is then set to 0 if the distance of its transformed vertex:

$$d^k(j) = |T^k\mathbf{b}_j - \mathbf{v}_j|$$

is greater than tol or e^k so that

$$w^{k+1}(j) = \begin{cases} w^0(j) = 1 \\ 1 \quad \text{if } d^k(j) < e^k \text{ or } d^k(j) < tol \\ 0 \quad \text{otherwise} \end{cases} \tag{5}$$

With this weighting function the algorithm will converge to

a value less than tol since it can be shown that, after the least squares minimisation,

$$\begin{aligned} e^{k+1} &= \min_T \sqrt{\textstyle\sum_{j=1}^{N'} w^{k+1}(j)|(T\mathbf{b}_j - \mathbf{v}_j^i)|^2 / \sum_{j=1}^{N'} w^{k+1}(j)} \\ &< \sqrt{\textstyle\sum_{j=1}^{N'} w^{k+1}(j)|(T^k\mathbf{b}_j - \mathbf{v}_j^i)|^2 / \sum_{j=1}^{N'} w^{k+1}(j)} \\ &< \sqrt{\max\{e^k, tol\}^2} = \max\{e^k, tol\}. \end{aligned}$$

So either the sequence of errors e^k is decreasing or it is less than tol and so has converged. Once convergence has been reached on frame i, the piece B_l^{i+1} is set to

$$B_l^{i+1} = \begin{cases} \{\mathbf{b}_j : w(j) = 1\} & \text{for the } L^2 \text{ norm (2)} \\ \{\mathbf{b}_j : d^k(j) > tol\} & \text{for the max norm (3)} \end{cases} \quad (6)$$

We then perform the same weighted least squares minimisation on frame $i+1$. Thus the number of vertices N' in each piece reduces as we progress through the frames.

The weighted least squares problem (4) becomes underdetermined if $\sum_{j=1}^{N'} w^{k+1}(j) < 3$. However this is also exactly the point at which RTs cease to become profitable for encoding the animation. This is because to encode two translations at each frame would take 6 floats and to encode a RT at each frame would also take 6 floats since there are 6 degrees of freedom in a rigid transformation. Therefore we choose to encode these pieces using translations (which are trivial to compute) and refer to them as "small pieces".

Once the weighted least squares problem (4, 5) has converged for each time frame we have determined a segment of the animation and set $B_l = B_l^M$. The process is then performed again for the unassigned vertices until all the vertices have been assigned. The base mesh is thus segmented into n independent pieces which each have a sequence of M transformations. The error of the approximation for each piece and hence for the whole approximation is within the prescribed tolerance. The whole process is outlined in the following pseudocode:

```
while(B_l^0 ≠ ∅ ){
    for(i = 1 to M ) {
      while (e_k > tol) {
          Solve weighted least squares minimisation 4 and 5
      }
      Set B_l^{i+1} using 6
    }
    B_l = B_l^M
    B_l^0 = {B − B_s for s = 0...l − 1}
}
```

6. Temporal Segmentation

Since each transformation has been computed for B_l^i and not for the final piece B_l we re-calculate the transformations again by solving 4 with weights fixed to 1.

The next step is to further exploit temporal coherence by aggregating RTs which are similar over time. To do this we compute the error of transforming a piece at time i with the previous transformation T_j^{i-1} instead of the current transformation T_j^i. If this error is bounded be tol then the transforms are aggregated, so that

$$\begin{aligned} \text{If} \quad & \textstyle\sum_{j=1}^{N'} |T_j^{i-1}\mathbf{b}_j - \mathbf{v}_j^i| < N' tol^2 \text{ or} \\ & \max'_{j=1\ldots N}\{T_j^{i-1}\mathbf{b}_j - \mathbf{v}_j^i\} < tol \\ \text{then} \quad & T_j^i = T_j^{i-1} \end{aligned}$$

depending on the error metric used. Thus the number of distinct RTs for each piece is reduced from M to m.

7. Smoothing

Although the method will give an accurate representation of the animation in a purely geometric sense, the animation may appear jerky in time and space. We have found that this is easily rectified with a simple spatial-temporal smoothing scheme. We first convert the transformations to a dense series of $M \times N$ translations $\delta\mathbf{u}_j^i$ which are the approximated version of $\delta\mathbf{v}_j^i$ in the representation 1.

The smoothing is performed by averaging over frames and neighboring vertices

$$\bar{\delta\mathbf{u}}_j^i = \frac{1}{\tau}\frac{1}{\sigma}\sum_{k=-\tau}^{\tau}\sum_{l=1}^{\sigma}\delta\mathbf{u}_{j+l}^{i+k}$$

where σ and τ are the spatial and temporal smoothing parameters respectively. The spatial summation is performed over the σ vertex rings surrounding the given vertex. The animation is reconstructed by translating $\mathbf{b}_j$ incrementally by the set of translations $\{\bar{\delta\mathbf{u}}_j^i\}$.

In practice we have found that a spatial smoothing parameter $\sigma = 1$ is usually sufficient. However, more temporal smoothing is required at lower levels of detail where transformations will be very sparse. We therefore vary τ between 0 for high levels of detail and 7 at low levels.

7.1. Compression and Error

We calculate the size of the compressed animations analytically by the formula

$$\begin{aligned} &\text{Size in Bits} = \\ &(\text{Big Piece Transforms} \times 6 + \text{Single Vertex Transforms} \times 3) \times Q \\ &+ \text{Transform Indices} + \text{Original Mesh} \end{aligned}$$

since an RT requires 6 floats to encode but a translation only requires 3. Transform indices are required to associate transformations with each vertex. The orginal mesh is encoded using Tuoma-Gotsman compression [13]. The quantisation level Q can be calculated as the smallest integer such that

$$Q > \log_2\left(\frac{f_{max} - f_{min}}{2\|e\|}\right)$$

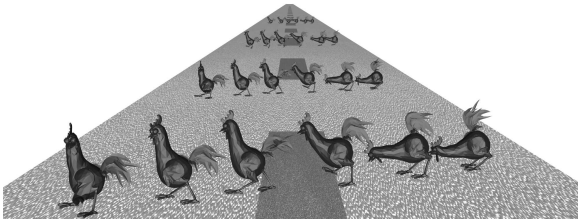

Figure 1: *Level of detail for chicken animation at* $tol = 0.01, 0.05, 0.1, 2$.

where f_{max} and f_{min} are the maximum and minimum floats to be encoded. This will ensure a quantisation error which is less than the error in the animation and hence will not affect the overall error greatly. Signal to noise ratio (SNR) is calculated as suggested in Khodakovsky [9] to be

$$\mathrm{SNR} = 20\log_{10}\frac{BoundingBoxDiagonal}{RMSE}$$

8. Results

We have tested the algorithm on several data sets which have been used to evaluate other schemes. We demonstrate segmentation of the mesh, level of detail, convergence, smooth animation and rate distortion analysis.

Figures (2, 8, 6) show the spatial segmentation of standard animation examples. It is clear that as the number of segments increases the animation converges to the original animation.

8.1. Convergence and Smoothed Convergence

Figure 9 shows the error over time for the snake example at different tolerances and smoothing. The error is bounded by tol for all times demonstrating convergence of the algorithm. We note that the smoothing algorithm does not greatly affect the error at low levels of detail. The error's quoted in this paper are for the smoothed animations. Figure 7 shows the effect of smoothing the animation.

8.2. Level of Detail

Figure 2 shows the chicken animation reconstructed at different levels of detail in the first 260 frames. At the lowest level of detail ($tol = 2$) there is only one piece which moves rigidly throughout the animation. Figure (1) shows the animations represented at various levels of detail.

8.3. Rate Distortion Analysis

Figures 3, 4 and 5 and tables 1, 2 and 3 show the rate-distortion analysis for the chicken, avatar and snake examples. For comparison the PCA algorithm of Alexa and Muller [1] was also implemented with the modification that the geometric basis was quantised to provide greater compression.

The tables show that the algorithm has converged successfully since all the errors are within the given tolerance. At lower tolerances the number of pieces and their transformations increases. We also show the result of the further temporal compression by noting the percentage of RTs which are kept after the aggregation (7). Both compression algorithms show an improvement on compressing each mesh separately using Tuoma-Gotsman compression. Also the tables show the possibility of compressing the animation below the animator's parameterisation for low levels of detail.

The rate distortion curves for the snake and avatar models show that our method outperforms that of the PCA method. In the avatar example (Figure 8) most of the animation is rigid. However, skeleton subspace deformation blends the rigid transforms of the bones so that near joints the animation is non-rigid. Interestingly the division is not as an animator may make it (at $tol = 0.1$ for example the left leg is segmented at the middle of the thigh rather than the knee). This shows that at this level of detail, the compression may be more efficient than the animator's parameterisation which would not necessarily find the most correlated pieces. For higher tolerances the scheme is seen to converge to the original animation. At $tol = 0.01$ we can see a large number of pieces are needed around the shoulder where much of the animation is non-rigid. The snake example shows the best rate-distortion performance and demonstrates how well non-rigid deformation can be approximated by rigid.

The chicken animation involves a character which is animating in an approximately rigid manner for the first 260 frames before doing a cartoonish "take" where the neck and eyes are stretched and inflated. This non-rigid "take" along with the animation of feathers and "cluckers" in the chicken example provides a difficult challenge for our scheme. Some of the difficulty is met by manually segmenting the animation into time sequences which are mostly rigid. Thus for $tol < 1$ we code the animation into different sequences (for example before, during and after the "take"). This method incurs an overhead of having to code a new base mesh geometry (the first mesh in each sequence) for each segment. This overhead is seen in the rate-distortion curve which shows that the purely geometric method of PCA performs better.

9. Conclusion

We have introduced piecewise rigid transformations as a basis for animation compression. We have used this basis to encode animations without having to encode any residuals. We have shown that this representation can encode an animation efficiently for visually relevant ranges of SNR. Furthermore, at low levels of detail, we have shown the possibility of compressing animated meshes beyond the compression given by the original animator's parameterisation.

Tol	Max Error	SNR	Big Pces	Small Pces	% RTs	Size (K)
1	0.853	12.4	6	0	13.4	37.2
0.5	0.462	17.7	20	2	20.3	42.7
0.1	0.1	31.0	113	48	55.3	129.9
0.01	0.0099	51.2	216	547	84.3	1085.8
Animator's Param. (kb)				307.2		
8 bit Tuoma Compression (kb)				1619.1		

Table 1: *Errors for compressing chicken (3030 vertices 400 frames)*

Tol	Max Error	SNR	Big Pces	Small Pces	% RTs	Size (K)
0.5	0.45	12.0	4	1	9.6	13.1
0.1	0.096	25.4	20	0	23.2	17.5
0.05	0.049	31.3	38	9	34.5	27.7
0.01	0.0094	45.6	63	68	67.7	93.0
Mesh + Mocap (kb)				27.8		
8 bit Tuoma Compression (kb)				357		

Table 2: *Errors for compressing avatar (1070 vertices 200 frames)*

From the rate-distortion analysis it would appear that the scheme performs most effectively with animations with some rigid components. However, the highly non-rigid movements in the chicken example prove hard to encode unless entirely new geometries are used. In the other examples small non-rigid animations are well approximated by our algorithm. We may conclude that prior knowledge of the type of animation to be compressed would enhance any scheme but that rigid transformations are a robust and cheap basis for most animations.

10. Further Work

This work may be extended in the following ways

- The method could be extended to include a hybrid of bases such as those suggested by Lengyel. A further level of optimisation would be required to select which basis to use.
- Further geometric compression is possible in the manner described by Shamir [12].
- An automatic method for deciding how to segment the animation over time would be more efficient. For example this method would compress the chicken "take" separately to the rest of the animation.
- Animations with different connectivity could be encoded with RTs once a correspondence between vertices has been established.

Tol	Max Error	SNR	Big Pces	Small Pces	% RTs	Size (K)
0.1	0.0585	18.8	3	0	7.22	5.92
0.02	0.0136	32.65	8	0	27.08	7.34
0.005	0.00491	46.96	56	10	74.18	45.56
0.001	0.000959	63.64	296	88	95.42	389.78
8 bit Tuoma Compression (kb)				584		

Table 3: *Errors for compressing snake (9179 vertices 121 frames)*

11. Acknowledgements

The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza, and Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques. Thanks go to Jed Lengyel for supplying this animation. This work has been supported by EPSRC grant "Functional Models" and by EU IST grant Melies. Matthias Muller

References

1. Marc Alexa and Wolfgang Müller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, August 2000. ISSN 1067-7055. 2, 2, 5
2. Hector M. Briceño, Pedro V. Sander, Leonard McMillan, Steven Gortler, and Hugues Hoppe. Geometry videos: A new representation fro 3d animations. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2003. 2, 2
3. Aaron Hertzmann Christoph Bregler and Henning Biermann. recovering non-rigid 3d shape from image streams. In *Proc. IEEE CVPR 2000*, 2000. 2
4. Douglas DeCarlo and Dimitris Metaxas. Deformable model-based shape and motion analysis from images using motion residual error. In *Proceedings ICCV '98*, pages 113–119, 1998. 2
5. Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. *ACM Transactions on Graphics*, 21(3):355–361, July 2002. 2
6. Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Frédéric Pighin. Making faces. *Proceedings of SIGGRAPH 98*, pages 55–66, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida. 2
7. Hugues Hoppe. Progressive meshes. *Proceedings of SIGGRAPH 96*, pages 99–108, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana. 2
8. Hugues H. Hoppe. New quadric metric for simplifying

meshes with appearance attributes. *IEEE Visualization '99*, pages 59–66, October 1999. ISBN 0-7803-5897-X. Held in San Francisco, California. 3

9. Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. Progressive geometry compression. *Proceedings of SIGGRAPH 2000*, pages 271–278, July 2000. ISBN 1-58113-208-5. 2, 5

10. Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pages 95–104, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida. 2

11. Jerome Edward Lengyel. Compression of time-dependent geometry. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 89–96. ACM SIGGRAPH, April 1999. ISBN 1-58113-082-1. 2

12. Ariel Shamir, Chandrajit L. Bajaj, and Valerio Pascucci. Multi-resolution dynamic meshes with arbitrary deformations. In *IEEE Visualization*, pages 423–430, 2000. 2, 6

13. Costa Touma and Craig Gotsman. Triangle mesh compression. In *Graphics Interface '98*, pages 26–34, June 1998. ISBN 0-9695338-6-1. 4

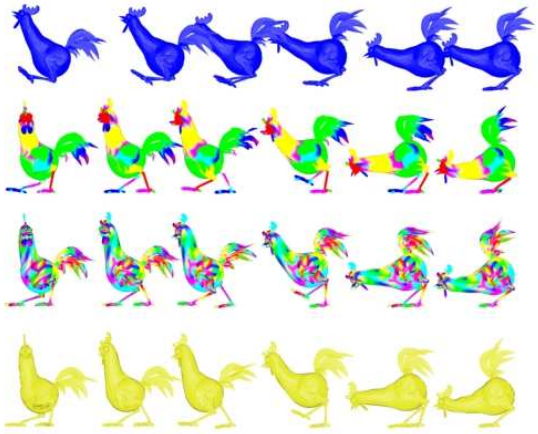

Figure 2: *Chicken animation at tolerance levels 2, 0.1, 0.01 and the original animation*

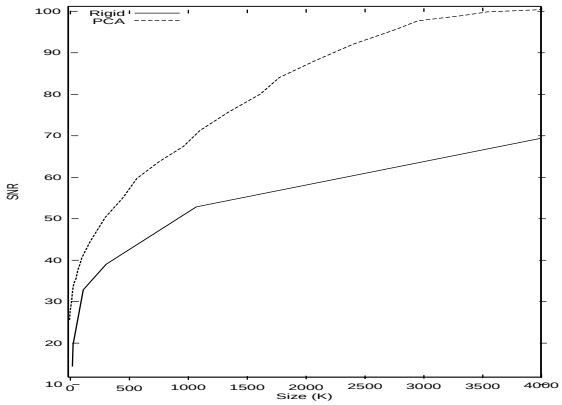

Figure 3: *Rate distortion curves for chicken animation*

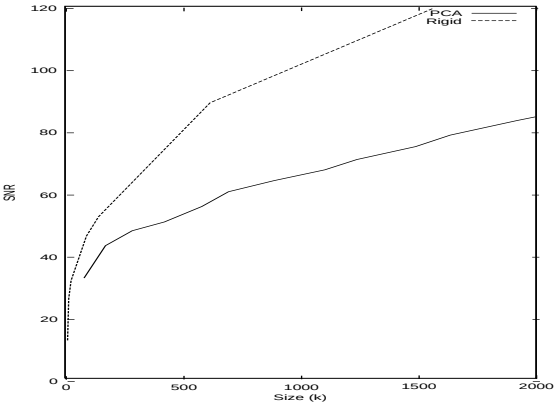

Figure 4: *Rate distortion curves for avatar animation*

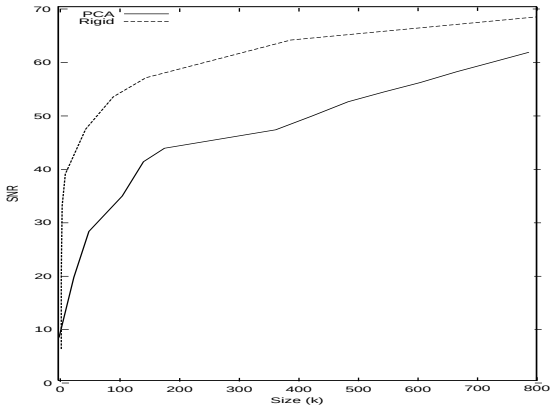

Figure 5: *Rate distortion curves for snake animation*

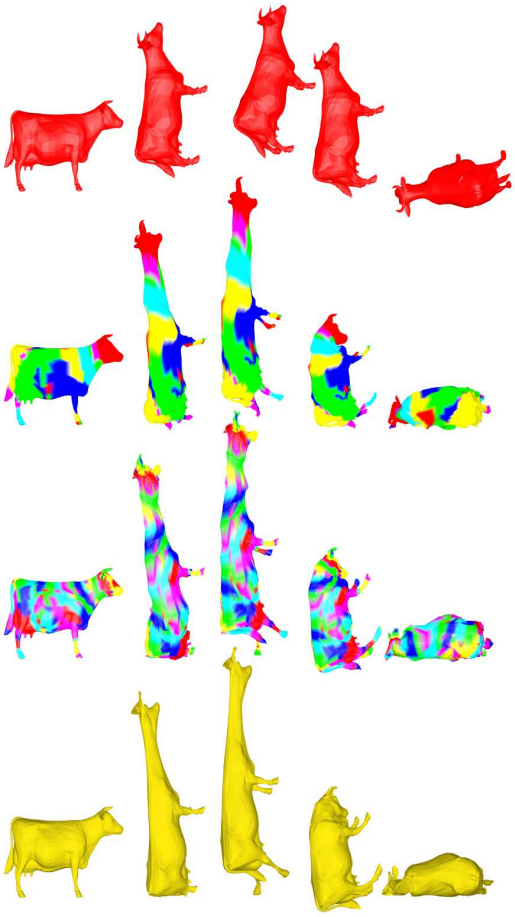

Figure 6: *Cow animation at tolerance levels 0.1, 0.02, 0.01 and the original animation*

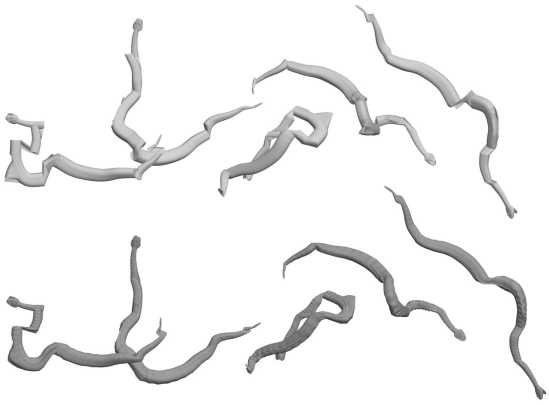

Figure 7: *Snake animation at* $tol = 0.02$ *without smoothing (top) and with smoothing* $\sigma = 1$ *and* $\tau = 3$

Figure 8: *Avatar animation at tolerance levels 0.5, 0.1 0.01 and the original animation*

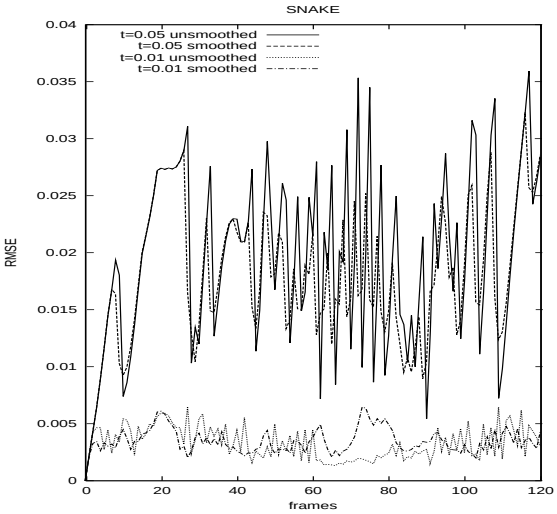

Figure 9: *Error over time for snake animation*