EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19, (2000), Number 3

Representing Animations by Principal Components

Marc Alexa and Wolfgang Müller

Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany

Abstract

In this paper, we present a representation for three-dimensional geometric animation sequences. Different from standard key-frame techniques, this approach is based on the determination of principal animation components and decouples the animation from the underlying geometry. The new representation supports progressive animation compression with spatial, as well as temporal, level-of-detail and high compression ratios. The distinction of animation and geometry allows for mapping animations onto other objects.

Keywords: Geometric animations, polyhedral morphing, multiresolution/progressive representation

1. Introduction

Computer animation has evolved into a standard technique in Computer Graphics. In the last few decades, a number of different animation techniques have been developed, but key-frame animation has established itself as the standard technique for describing time-dependent 3D Computer Animation. Instead of describing every single frame, only a sequence of principal frames - so called key-frames - is defined and additional frames are generated by interpolating between two consecutive key-frames using in-betweening.

Elaborated techniques have been developed to allow for the automated generation of physically-based behavior, namely kinematics and inverse kinematics, and today animation systems are increasingly coupled with simulation engines to facilitate the production of complex and realistic animations. However, key-frames remain the standard for representing animations.

Despite its widespread use, this concept of merging object and geometry descriptions has a fundamental problem. While it is easy to specify, design, or output an animation in terms of key-frames, it is difficult to manage due to the large amount of data or to change the time-behavior since all meta-information is lost. In particular, the following problems connected to geometric key-frame animations can be stated:

- Redundancy: A complete object description has to be recorded for each key-frame, even if parts of the object do not change at all. Additionally, repetitive patterns result in repeating the geometry description. Consequently, animation sequences are usually very large and hard to apply in streaming applications. The compression of such sequences is a problem which is under intense investigation in the Computer Graphics research community today.
- Modification: Exchanging an animated object in a scene while reusing the once specified animation at the same time is an involved task, even though most of the necessary information should be available. In general, after introducing the new model, it has to be animated again by hand. Similarly, it is almost impossible to make use of a once defined animation and to extend or exchange the object's behavior.
  The aspect of reuse has been addressed for specific object domains. One approach is standardized parameterization of an object and its possible behaviors, thus allowing for the exchange of both geometry and animation (e.g. Humanoid Animation[26] or Facial Animation[20] in MPEG-4). Similar, but more general, is the idea of animation elements[10], general object hierarchies with a defined interface. However, both approaches do not provide a general solution to this problem.
- Level of detail: Another problem of great consequence is the reduction of scene complexity in interactive applications. LOD concepts, such as progressive meshes[16], allow static objects to be fitted to the display requirements. Recent techniques try to provide a view-dependent level of detail for static objects based on mesh-simplification techniques or sometimes by

exploiting progressive transmission and decompression schemes.

However, if the objects are animated, these standard techniques may fail or not be applicable at all. Standard LOD hierarchies could not be applied for animation, since hierarchies had to be provided for all key-frames.

Still, geometric simplification exploits just *spatial* coherence, while animations additionally exhibit *temporal* coherence. Surprisingly, the application of an LOD concept for animated geometry and animation itself has not been discussed in the literature to the best of our knowledge. For the same reasons small, static geometry features are omitted in standard LOD techniques, small temporal features should also be a target of LOD approaches.

We present an alternative representation of animation sequences based on principal animation components, which alleviates the above-mentioned problems.

The remainder of this paper is organized as follows: In the following chapter, a discussion of relevant work in the area of object spaces is given. After this, we introduce the fundamentals of our *principal component representation* for animations and the construction of such representations using a *Principal Component Analysis*. Finally, we present selected application examples and discuss the results of our paper in some detail.

2. Related Work

Our work touches on existing approaches in various fields. Most important are the areas of mesh compression and simplification, as well as shape spaces, which describe families of shapes as a linear space.

Most work on mesh simplification and compression has concentrated on static geometry. In a recent survey, Garland[12] reports that "all current simplification methods assume that the surface being simplified is rigid". Naturally, mesh simplification techniques lead to a hierarchical representation of meshes, also known as *progressive meshes*[16]. Recent techniques, however, compress the mesh and its hierarchy. Cohen-Or et al. report compression ratios comparable to the best known for static meshes (e.g. Touma and Gotsman[24], Taubin et al.[23], Gumhold and Strasser[15], Rossignac[21]) and their technique allows for streaming a mesh progressively.

The majority of today's geometry compression techniques are based on prediction. The difference between predicted and actual locations is usually small and can be coded using only few bits. Prediction-based compression appraoches have been extended for time-dependent geometry by Lengyel[19]. In this work, vertices are split into sets and for each set an affine transform describing the vertex paths is approximated.

One of the main ideas of this paper is to structure the elements of an animated scene in terms of a linear space. This idea is not new. Edelsbrunner[11] appears to have first mentioned the term *shape space* and it was explicitly used by Cheng, Edelsbrunner, and Fu[6] for interpolating between implicit descriptions of shapes. Alexa and Müller derive spaces from any morphing technique[1] and investigate conditions under which these spaces are linear. The idea of candidate spaces for object recognition was used by Turk and Pentland[25]. The idea is to construct typical faces from a set of photographs, which is done by computing the Eigenvectors of the space over all photographs (vectors of gray-levels). A space of three dimensional-shapes for recognition and modeling of faces was used by Blanz and Vetter[5].

In this work, we derive linear spaces by performing a basis transformation for appropriately represented animated polyhedral models. The resulting spaces can be used to describe animation sequences in terms of principle components and their influence over time. The abstraction allows for changes in the geometry as in the animation without affecting each other. We can, for instance, use the same animation with different geometries. Morevoer, interesting special effects may be achieved by exchanging single base elements of the linear space. These and other applications are demonstrated in Section 5.

3. The idea

In this paper, we restrict ourselves to the discussion of scenes comprising animated polyhedral shapes described by key-frame geometries B_i. All polyhedral shapes are assumed to have an isomorphic vertex-edge topology T. Each shape B_i is defined in terms of attributes of its boundary elements. Typically, at least a coordinate is associated to a vertex, but other attributes such as a normal, a color value, a texture coordinate, etc might be linked to it. Also, other boundary elements may have attributes, e.g. normals might be associated to faces rather than vertices. In general, vertex attributes can be described by a vector of scalars and also all vertices of a frame B_i can be represented as a vector. We assume that all base shapes have vectors of same length and that the attributes are arranged identically. In the following, B_i is used equivalently with the vector of these attributes.

The state of an object in a key-frame animation can then be calculated by interpolating between two consecutive key-frames. Formally, this can be described as:

$$A(t) = \sum_i a_i(t) \cdot B_i \qquad (1)$$

where $A(t)$ stands for the object's state at time t and $a_i(t)$ are the weights describing the key-frame interpolation, i.e.

$$a_i(t) = \left(0, \ldots, 0, \frac{t_{i+1} - t}{t_{i+1} - t_i}, \frac{t - t_i}{t_{i+1} - t_i}, 0, \ldots, 0\right)$$

where t_i is the time stamp of the i-th key-frame.

However, if we want to separate geometry from animation, an alternative representation would be useful, such as:

$$A(t) = \tilde{a}_0(t) \cdot \tilde{B}_0 + \sum_i \tilde{a}_i(t) \cdot \tilde{B}_i \qquad (2)$$

where $\tilde{B}_0$ represents the average geometry of the shape and the sum describes deviations from this representative geometry. We illustrate this at the example of a facial animation: All key-frames B_i of such an animation would describe the geometry of a face. In the alternative representation of (2), $\tilde{B}_0$ would be the geometry of a face, and all other $\tilde{B}_i$ would describe differences to that face necessary to describe the animation. An example in section 5 shows why and how this helps in decoupling animation and geometry.

This process of extracting a base component in the representation could be repeated in order to find the principal deviation from the average geometry, i.e. a geometry which is the average difference of $\tilde{B}_0$ and the animation sequence. Generally, it makes sense to sort the geometries based on their geometric importance. We denote this description as

$$A(t) = \sum_i \hat{a}_i(t) \cdot \hat{B}_i \ . \qquad (3)$$

where $\hat{B}_0$ represents the average static geometry while the remaining factors $\hat{B}_i$ represent geometric changes with decreasing importance with respect to the reconstruction of the animation. Note that representations (1), (2), and (3) are just basis transforms of each other, i.e. a matrix multiplication transforms one into another.

However, simple rigid motion of the object may render this approach obsolete because the linear deviations $\hat{B}_i$ from the base geometry $\hat{B}_0$ cannot include e.g. rotations (see also Lengyel[19]). For this reason, we propose to decompose the animation into rigid body motion and an elastic part first. This idea has been successfully applied in geometric morphing techniques[7]. In this context, we proceed as follows: First, all shapes are translated so that their center of mass coincides with the origin. Then, an affine map is computed minimizing the squared distance of corresponding vertices with regard to the first frame. The affine map is restricted to matrix representations with determinants greater zero, since reflections do not seem appropriate and the matrix has to be invertible. Results of this approach are depicted in Figure 1.

If we assume our model to be represented in homogenous coordinates, we can write the necessary transformation as a single matrix multiplication. That is, instead of the key-frames B_i we use transformed key-frames $T(t_i) \cdot B_i$. This has to be taken into account when the animation is reconstructed. Here, we have to use a slightly different representation:

Figure 1: *The normalization step. The upper row shows frames from a chicken animation translated so that the center of mass coincides with the origin. The lower row shows transformed shapes, where the squared distance of corresponding vertices is minimized.*

$$A(t) = T^{-1}(t) \cdot \sum_i a_i(t) \cdot \hat{B}_i \qquad (4)$$

Assuming the B_i behave as described above, we have a representation where animation and geometry are clearly decoupled. The geometry part is described mainly by $\hat{B}_0$, while the main animation is described by the T_i and $a_i(t)$. $\hat{B}_1$, $\hat{B}_2$, $\hat{B}_3$, and so on describe all possible deviations of the geometry $\hat{B}_0$.

This representation of the animation sequence makes it easy to perform compression and LOD operations. By restricting the representation to the first few components $\hat{B}_i$, high compression ratios can be achieved while omitting only unimportant features. Furthermore, metric LOD techniques are better supported since progressive meshes have to be generated and held in memory for these few components only.

At the same time, the number of bases used in (4) affects the accuracy of the animation. Few $\hat{B}_i$ result in a coarse representation of the animation, more $\hat{B}_i$ components in higher accuracy. Thus, this representation is inherently progressive.

Further implications of this representation are shown and discussed in sections 5 and 6. In the upcoming section, we explain how to find the above description.

4. Principal Component Analysis

A process of analyzing the relationship between base vectors of a space is the *Principal Component Analysis* (PCA). In our scenario, the PCA first determines the average shape that contains the common properties of the shapes in all key-frames. Other components will represent differences to this shape. This is also interesting when it comes to coding

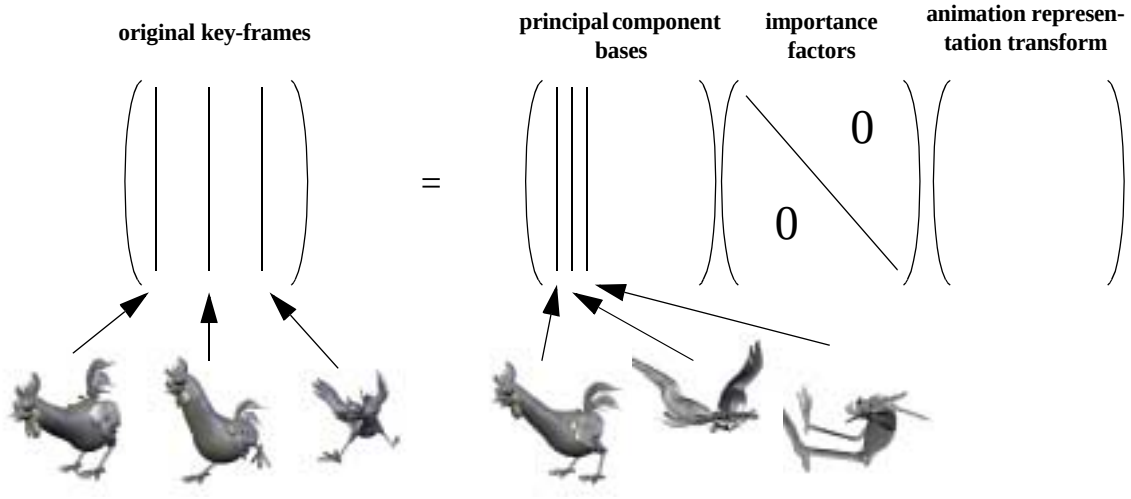

Figure 2: *The Principal Component Analysis for geometric animations illustrated.*

the bases of a shape, as it exploits similarity and turns it into zeros, which are easily compressed using entropy encoding.

There are several ways of finding principal components. In our case, we are not only concerned with finding the most important principal components to give a rough approximation of the shapes. In addition, we want to find an alternative basis and cut only a few non-contributing vectors. A way of finding this basis is the singular value decomposition (SVD[13]). The SVD decomposes a real matrix into an orthogonal, a diagonal, and an orthogonal one.

Formally, we can write the non-rigid part of the original key-frames in matrix form:

$$B = (T_0 B_0, T_1 B_1, \ldots, T_{n-1} B_{n-1}) \tag{5}$$

Using the SVD, we find the following:

$$B = \hat{B} \cdot S \cdot V^T \tag{6}$$

The values of the diagonal matrix S are the singular values. The closer a singular value is to zero, the closer a base shape is to being linear-dependent. The first orthogonal matrix $\hat{B}$ contains the basis of the space with base vectors corresponding to the singular values, i.e. the rows contain the $\hat{B}_i$ we are searching for. The matrix representation and SVD are visualized in Figure 1.

A severe problem with the SVD is that it is very costly and likely to reach its limits on modern computers when applied to matrices with the size of the vertex count of typical models times the number of key-frames. A solution is to simplify the base shapes and to not consider every key-frame. This is rectified by the spatial and temporal coherence typically exhibited in geometric animations. In particular, it is in many cases sufficient to consider only every second up to every fifth key-frame. However, adaptive schemes for the selection of key-frames to consider would be desirable.

The representation vectors a_i for key-frames which have not been considered for the SVD can be obtained by projecting the key-frame into the new basis. Since the basis constructed with the SVD is orthonormal, computing inner products of the key-frames and the new base vectors is the desired projection.

5. Applications and Results

In this section, we present results of computing a PCA for two animation sequences. We show how the PCA leads to a compressed progressive representation and fosters the exchange of geometry or behavior.

The first example is a part of the *Chicken Crossing* animation, in particular, 400 frames of the chicken's geometry. The sequence is highly non-linear, i.e. it comprises rigid body motion and dynamic soft body changes. The geometry consists of 3030 vertices. Disregarding the topology information, this results in an uncompressed size of 400 frames x 3030 vertices x 3 dimensions x 4 byte = 14,544,000 bytes.

To generate the principal component representation we first normalized the frames using the linear least squares fit. The resulting key-frames were composed into a 9090x400 matrix and a SVD was performed. The resulting orthogonal matrix was used to define the new base vectors replacing the key-frames.

We reconstructed the animated sequences using different numbers of base objects. This was done by setting appropriate singular values to zero. The results are shown in Figure

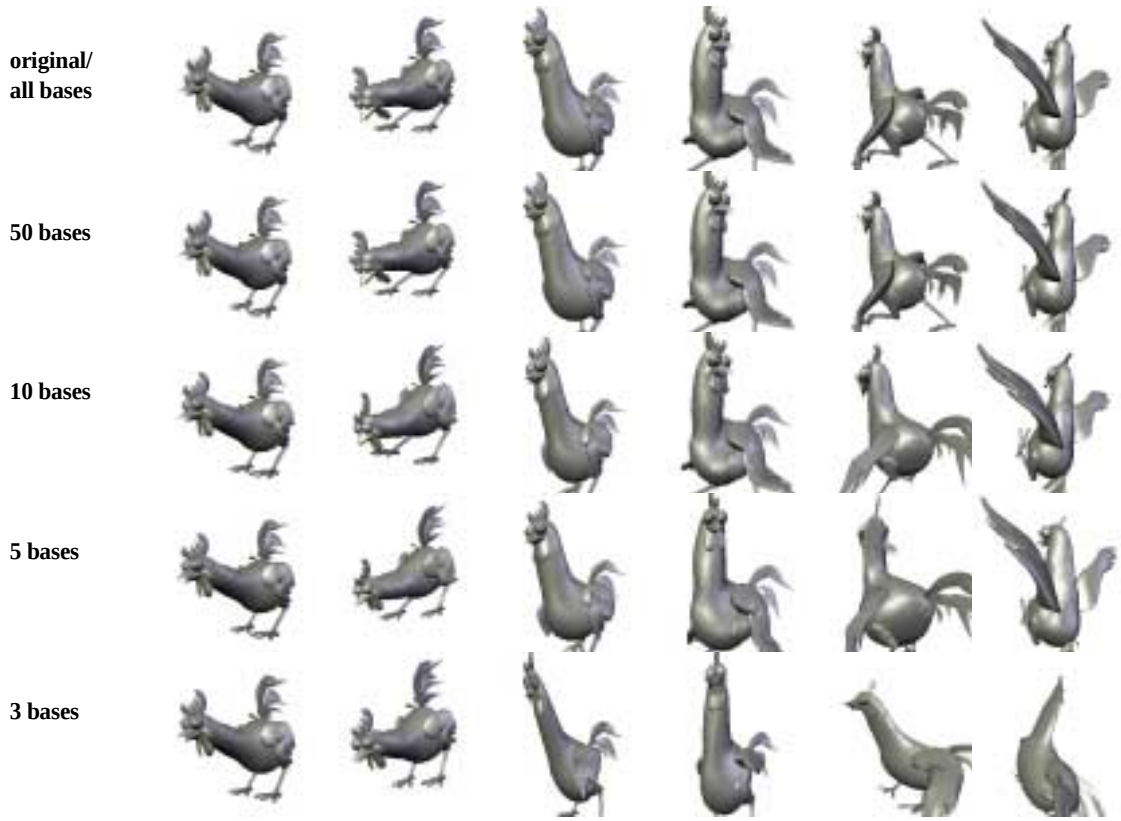

Figure 3: *Principal Component Analysis applied to the chicken animation. Prior to the SVD, base shapes are normalized. The sequences above show frames 0, 80, 160, 240, 320, and 400 using different number of bases in the reconstruction.*

3 and in the accompanying videos. The animation still looks very reasonable with only 10 base shapes. Even the animation using only 5 bases can be used if viewed from far away. This would correspond to the definition of a complex LOD.

encoding	size	ratio
original	14,544,000	
all base shapes (lossless)	14,548,800	1:0.99
50 base shapes (lossy)	1,818,600	1:8.3
10 base shapes (lossy)	364,800	1:39.8
5 base shapes (lossy)	181,860	1:79.97
3 base shapes (lossy)	109,116	1:133.28

Table 1: *Compression ratios for principal component representation. Note that sizes and ratios include the additional costs for storing the transformation matrices from the normalization step.*

By omitting a number of the unimportant base components, very high compression ratios can be achieved. Table 1 shows the compression ratios for the animation. However, this animation is by far not typical. It comprises a highly deformable object. For typical animation sequences, even better results can be expected.

In the second example we applied the principal component representation in a facial animation system. This system uses several facial expressions coded as polyhedral models with isomorphic vertex-edge topology. Key-frames are generated by blending several expressions (e.g. saying an 'A' and smiling). In this system, the animation is represented as a vector over time that describes the linear combination of base shapes. Thus, animation representation and geometry are already decoupled in this specific application. In this example, we show how the geometry of the avatar can be exchanged with another geometry making use of the already existing animation descriptions.

Feature-based polyhedral morphing[2,14,18] aims at finding correspondences between different polyhedral models. Most of these techniques rely on topological merging[17], i.e. a vertex edge topology is produced that contains both origi-

Figure 4: *Exchanging geometry and in existing animations. a) A facial animation defined by a linear combination of base shapes. b) A featured-guided morph between the original avatar mesh and a new mesh. Topological merging is used to produce a mesh which represents both shapes. Feature control assures that the same vertices represent common features (e.g. mouth, eyes, etc.) c) The new mesh can be used with the existing animation with no additional user intervention. d) The morph can even be applied while the animation is performed.*

nal vertex-edge topologies as subgraphs. We have used a feature-based morphing technique to produce a mesh that can represent both the original avatar and another face. By defining a few vertex-vertex correspondences, we make sure that same vertices represent common features in the source and the target model. This results in the convenient fact that we can combine the new face with expressions of the old one, e.g. we can add the smile defined in terms of the original avatar to the new face. This means, by defining the correspondences between the two neutral faces, we get all other expressions of the face automatically. Play-back animations authored for the original avatar can be directly applied to the new face. In addition, we can morph between the two faces while the animation is performed, since the avatar and the new face now share the same vertex edge-topology. This is an impressive example of the effectiveness of decoupling animation representation from geometry. Results are depicted in Figure 4 and in the accompanying videos.

6. Conclusions and Outlook

We present an approach for representing animation sequences based on the principal components of key-frame geometries. The principal component representation allows for an easy and adaptive lossy compression of animation sequences with factors up to 1:100 accepting loss in animation accuracy. It would be interesting to quantify that loss relative to the compression ratio. However, measures of geometric deformation seem to be a topic of current

research and not yet applicable. Moreover, standard compression techniques were not exploited at all in this calculation. Additional compression can be achieved by compressing the base shape matrices. Again, the principal component analysis reorganizes the base shapes so that bases with a higher index will contain more zeros. This could be exploited with an additional entropy encoding.

The order of base objects naturally supports progressive transmission of the animation base shapes. At the same time, the inherent hierarchy could be used for LOD techniques. This could go hand in hand with a progressive mesh. The importance ordering of base shapes additionally eases mesh simplification since only the geometric features of a few meshes have to be taken into account for simplification. In experiments we found that standard simplification techniques can be extended to handle more than one mesh.

The animation itself is represented by small vectors if the number of necessary base shapes is small. Note that the number of necessary base shapes is bound by the number of key frames. While the number of base shapes in our examples is much less than the number of key frames, it might be useful to break very long animation sequences into pieces. It would be interesting to exchange only parts of a base while streaming an animation.

Having small vectors represent the animation allows for streaming over virtually every network in real time. The decoupling of animation and geometry enables managing and changing animations, e.g. exchanging the animated object according to the client's display capabilities. Mapping existing animations to a new object offers new ways of authoring.

The necessary operations to play back animations defined as linear combinations of a small set of base shapes are now part of the proposals for animation in MPEG4.

Acknowledgements

This work would not exist without Herbert Edelsbrunner, who introduced the first author to the idea of shape spaces. Part of this work was done when the first author was visiting Tel-Aviv University supported by the Hermann Minkowski - Minerva Center for Geometry. Thanks to Daniel Cohen-Or, David Levin, and Craig Gotsman for fruitful discussions.

We also want to thank Ulrike Spierling and Manfred Gaida for providing the avatar models and Jed Lengyel for making the chicken sequence public. 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.

References

1. Marc Alexa and Wolfgang Müller. The Morphing Space. Proceedings of WSCG '99, pp. 329-336, 1999
2. Marc Alexa. Merging Polyhedral Shapes with Scattered Features. Proceedings of Shape Modeling International '99, 1999, refined version in The Visual Computer, 16, 1, pp. 26-37, 2000
3. Marc Alexa, Johannes Behr, and Wolfgang Müller. The MorphNode. Proceedings of VRML/Web3D 2000, Monterey, CA, 2000
4. Mark de Berg, Mark van Krefeld, Mark Overmars, and Otfried Schwarzkopf. Computational Geometry - Algorithms and Applications. Springer, Berlin, 1997
5. Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. SIGGRAPH '99 Proceedings, pp. 187-194, 1999
6. H. Cheng, Herbert Edelsbrunner, and Ping Fu. Shape Space from Deformation. Manuscript, 1997
7. Daniel Cohen-Or, David Levin, and Amira Solomovici. Three dimensional distance field metamorphosis. ACM Transactions on Graphics, 1998
8. Daniel Cohen-Or, David Levin, Offir Remez. Progressive Compression of Arbitrary Triangular Meshes. Proceedings of Visualization '99, 1999
9. Michael F. Deering. Geometry Compression. SIGGRAPH '95 Proceedings, pp. 13-20, 1995
10. R. Dörner, V. Luckas, and U. Spierling. Ubiquitous Animation - an Element-Based Concept to Make 3D Animations Commonplace. In: Visual Proceedings SIGGRAPH '97, Los Angeles, CA, ACM Press 1997
11. Herbert Edelsbrunner. Deformable Smooth Surface Design. Report rgi-96-002, Raindrop Geomagic, Inc., 1996
12. Michael Garland. Multiresolution Modeling: Survey & Future Opportunities. Eurographics '99 State of the Art Report, pp. 111-131, 1999
13. Gene H. Golub and Charles F. van Loan. Matrix Computations. The Johns Hopkins University Press, Baltimore, 1983
14. A. Greogory, A. State, M. Lin, D. Manocha, and M. Livingston. Feature-based surface decomposition for correspondence and morphing between polyhedra. Proceedings of Computer Animation '98, pp. 64-71, 1998
15. Stefan Gumhold and Wolfgang Straßer. Real Time Compression of Triangle Mesh Connectivity. SIGGRAPH '98 Proceedings, pp. 133-140, 1998

16. Hugues Hoppe. Progressive Meshes. SIGGRAPH '96 Proceedings, pp. 99-108, 1996

17. Jim R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape Transformation for Polyhedral Objects. Computer Graphics, 26, pp. 47-54, 1992

18. Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution Mesh Morphing. SIGGRAPH '99 Proceedings, pp. 343-350, 1999

19. J.E. Lengyel. Compression of Time-Dependent Geometry, ACM Symposium on Interactive 3D Graphics, Atlanta, 1999

20. J. Ostermann. Animation of Synthetic Faces in MPEG-4. Computer Animation, pp. 49-51, Philadelphia, Pennsylvania, 1998

21. Jarek Rossignac. Edgebreaker. IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 1, 1999

22. Ken Shoemake and T. Duff. Matrix Animation and Polar Decomposition. Proceedings of Graphics Interface '92, pp. 258-264, 1992

23. Gabriel Taubin, Andre Gueziec, William Horn, and Francis Lazarus. Progressive forest split compression. SIGGRAPH 98 Proceedings, pp. 123-132, 1998

24. C. Touma and C. Gotsman. Triangle Mesh Compression. Graphics Interface ‘98, pp. 26-34, 1998

25. M.A. Turk, A.P. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1), pp. 71-86, 1991

26. Web3D Consortium Humanoid Animation Working Group (H-ANIM): http:// ece.uwaterloo.ca:80/~h-anim/, 1999