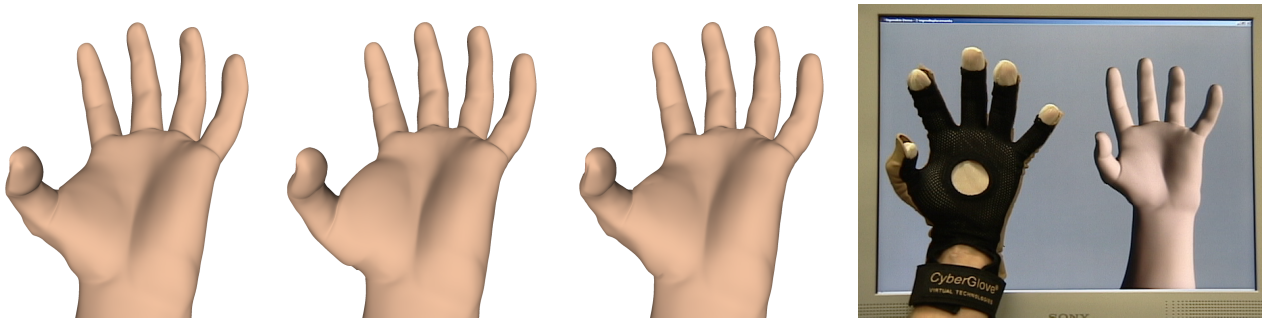
EigenSkin: Real Time Large Deformation Character Skinning in Hardware

Paul G. Kry, Doug L. James, and Dinesh K. Pai
University of British Columbia
{pgkry|djames|pai}@cs.ubc.ca



a) FEM simulated pose b) SSD only c) EigenSkin d) Real time hardware simulation

Figure 1: Comparison of EigenSkin and Skeletal-Subspace Deformation for an extreme pose not in the training data. Note significant differences in the thumb between a) the new pose computed from our finite element hand model, b) skeletal-subspace deformation only, and c) EigenSkin with one eigendisplacements and one normal correction per support. Figure d) shows our EigenSkin hand example being animated using a CyberGlove. The hand model shown here consists of 55,904 triangles and is drawn using display lists with a GeForce3 vertex program.

Abstract

We present a technique which allows subtle nonlinear quasi-static deformations of articulated characters to be compactly approximated by data-dependent eigenbases which are optimized for real time rendering on commodity graphics hardware. The method extends the common Skeletal-Subspace Deformation (SSD) technique to provide efficient approximations of the complex deformation behaviours exhibited in simulated, measured, and artist-drawn characters. Instead of storing displacements for key poses (which may be numerous), we precompute principal components of the deformation influences for individual kinematic joints, and so construct error-optimal eigenbases describing each joint's deformation subspace. Pose-dependent deformations are then expressed in terms of these reduced eigenbases, allowing precomputed coefficients of the eigenbasis to be interpolated at run time. Vertex program hardware can then efficiently render nonlinear skin deformations using a small number of eigendisplacements stored in graphics hardware. We refer to the final resulting character skinning construct as the model's *EigenSkin.* Animation results are presented for a very large nonlinear finite element model of a human hand rendered in real time at minimal cost to the main CPU.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically-based modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation, Virtual reality

Keywords: skeletal-subspace deformation, pose-space deformation, principal component analysis, hardware rendering

1 Introduction

Rendering of complex physical deformation models for character animation remains a significant hurdle for interactive applications, but one that has been largely overcome for off-line animation. Currently, most real time character animation, e.g., for video games, is done using a very common linear transform blending technique called (among other things) Skeletal-Subspace Deformation (SSD) [Magnenat-Thalmann et al. 1988]. It is extremely popular for its simplicity and plausibility, and is also widely supported by graphics hardware accelerators. Despite this, it is also widely known to suffer from several key problems:

- buckling of skin near joints, e.g., elbows, in extreme poses;
- poor behaviour near more complicated joints, such as shoulders and thumbs;
- restrictions on the range of deformations that can be easily modeled and displayed for various character poses.

While methods have been proposed to address this and have been effectively employed by the motion picture industry [Lewis et al. 2000], due to memory and graphics hardware constraints nearly all video game character animation is still done using traditional SSD.

In this paper, we present a practical technique which overcomes all aforementioned SSD problems, and can be achieved using a

memory-efficient linear correction to the traditional SSD method. The resulting *EigenSkin* construct allows subtle character deformations for skin and clothing, such as those derived from highly realistic artist-drawn poses, measurements from the real world, or laboriously computed anatomically and physically-based models. The deformations can be compactly represented in an efficient data-dependent basis and rendered in real time using vertex shaders in commodity graphics hardware, e.g., see [Lindholm et al. 2001].

Our approach is to start with an artist's SSD approximation of the character in question, as well as with geometry corresponding to particular key poses not well approximated by SSD. Vertex displacements between a given pose and the SSD model are mapped back to the neutral character pose, providing a displacement field pose correction. Instead of storing these displacement fields for each key pose and then interpolating between them at runtime, as in Pose Space Deformation (PSD) [Lewis et al. 2000], we use Principal Component Analysis (PCA) to construct an error-optimal eigendisplacement basis for representing this potentially large set of pose corrections. However, we do not simply use PCA on the displacement field defined over the entire surface, since this would lead to a large number of important basis functions and be inefficient for hardware rendering. Instead, we decompose the model into locally supported domains learned from the influence of individual joints on the displacement fields (described in detail in Section 2.2). The resulting memory sensitive set of locally supported eigendisplacement basis functions constitutes the EigenSkin approximation, and is well suited to rendering in graphics hardware. Please see Figure 1 for an example of EigenSkin results.

1.1 Previous Work

Significant work has occurred in graphics for deforming articulated characters using geometric methods [Magnenat-Thalmann et al. 1988; Singh and Kokkevis 2000; Lewis et al. 2000] and physically-based methods [Wilhelms and van Gelder 1997; Scheepers et al. 1997; Gourret et al. 1989]. Despite this, most character animation in interactive applications, such as video games, is based on a geometric skeletal deformation technique commonly referred to as linear blending, or matrix palette skinning, or Skeletal-Subspace Deformation (SSD), in which vertex locations are weighted averages of points in several coordinate frames (see [Magnenat-Thalmann et al. 1988; Magnenat-Thalmann and Thalmann 1991]).

One alternative is to store a large database of character poses, and interpolate between them [Maestri 1999]. While these approaches give animators great control over character deformation, they have the disadvantage of requiring a potentially very large number of poses for animation, and also lack an underlying kinematic model. Nevertheless, such approaches are common, especially for facial animation [Parke et al. 1996].

A hybrid approach which effectively combines SSD and morphing, is the work of Lewis et al. who introduced "Pose Space Deformations" (PSD) [Lewis et al. 2000] to overcome the limitations of linear transform blending while retaining a kinematic approach. Starting with a (simple) SSD model, they then store vertex displacement offsets between the SSD surface and various character poses. At run time, the character may be simulated by mapping interpolated displacements onto the underlying SSD character model, thereby providing a kinematic deformation model which also has artist-drawn poses. While this is a big improvement over character morphing, and sufficiently interactive for animators, storing surface displacements for each pose in a large pose space is a memory inefficient approach for hardware applications.

Similar to PSD, Sloan et al. show a more efficient method of interpolating an articulated figure using example shapes scattered in an abstract space [Sloan et al. 2001]. The abstract space consists of dimensions describing global properties of the shape, such as age and gender, but also includes dimensions used to describe configuration, such as the amount of bend at an elbow. Like our method, interpolation occurs in the rest pose before SSD is applied, however, the interpolation involves blending over all of the example shapes for every vertex. This becomes inefficient and difficult to map to hardware with the large number of examples required for a highly articulated figure since the independence of abstract space dimensions is not taken into account (e.g., bend in left elbow and bend in right elbow).

In addition to character poses created by 3D artists, we also wish to efficiently render deformation behaviour computed using physically-based and reality-based deformable models. Such models have been widely used [Terzopoulos and Fleischer 1988; Terzopoulos and Witkin 1988; Metaxas and Terzopoulos 1992; Cani-Gascuel 1998; O'Brien and Hodgins 1999; Pai et al. 2001; Allen et al. 2002], although most approaches are not intended for real time (hardware) rendering. Recently, approaches for fast simulation of physical dynamic volumetric deformations have appeared [Zhuang and Canny 1999; Debunne et al. 2001; Picinbono et al. 2001] for interactive applications, such as surgical simulation. Our interest is more closely related to quasi-static deformation, for which fast deformation techniques also exist [Cotin et al. 1999; James and Pai 1999] but are unfortunately restricted to small deformations unlike those associated with articulated characters (although see [James and Pai 2002b]). More closely related to character animation is *anatomically based modeling* of physical deformable models [Wilhelms and van Gelder 1997]; examples include musculature [Chen and Zeltzer 1992; Scheepers et al. 1997] and faces [Lee et al. 1995].

We note that a large class of pose-dependent quasi-static deformations can be described using the EigenSkin approach, largely independent of their origin, whether artist-drawn, measured, or anatomically based physical models. For example, pose-space parameterization of nonhysteretic cloth on articulated characters has recently been considered [Herman 2001], and could be optimized for hardware rendering using the techniques presented herein.

Finally, the use of reduced eigenbasis representations for high-dimensional models has a long history in science, with foundations on Principal Component Analysis and Karhunen-Loeve theory [Jolliffe 1986; Hyvarinen et al. 2001]. Related deformation topics include a morphable model for face synthesis [Blanz and Vetter 1999], modal analysis for dynamic vibrations [Pentland and Williams July 1989; James and Pai 2002a], decomposition of static deformations [Bookstein 1989], and recognition applications in computer vision, e.g., face recognition [Turk and Pentland 1991].

1.2 Our Contribution

We introduce a method for extending SSD that enhances its range of modeling capabilities at very little cost, and in a manner optimized for real time graphics hardware. EigenSkin constitutes an error-optimal set of eigenbases for approximating the original deformation model, for a given amount of per-vertex displacement memory. We illustrate our method by rendering a very large finite element model (which took several hundred hours to compute) at interactive rates on a PC with *negligible cost to the main CPU*. Using commodity graphics hardware, EigenSkin enables the simulation of subtle nonlinear surface deformations of geometrically complex models at little more than the cost of rendering.

2 Method

In this section we describe the process of augmenting an existing SSD model with EigenSkin. Although the process is shown for displacements, it applies similarly to the construction of linear normal corrections, allowing EigenSkin to correct SSD for both shape and shading.

2.1 Notation: SSD and Bone Weights

Let $\mathscr{B}$ be the set of all bone indices, and denote the bones affecting vertex i by the subset of indices $B_i \subset \mathscr{B}$. For a given skeletal configuration, with bone transforms $\{T_b\}_{b\in\mathscr{B}}$, the position of the i^{th} vertex after SSD is

$$\tilde{v}_i = \left(\sum_{b\in B_i} w_{ib} T_b\right) v_i \qquad (1)$$

where v_i is the position of vertex i in the neutral pose, and w_{ib} give the affine combination of bone transforms for this vertex. In the character's neutral pose we assume that $T_b = I, \forall b \in \mathscr{B}$.

Starting with a reasonable set of bone weights is important because the added displacements only correct the SSD predicted mesh shape near observed configurations. We compute our SSD bone weights as a function of vertex bone distances in the neutral pose. This yields reasonable bone weights which change smoothly over the mesh. Filtering may be required to force each bone's weights to zero at the edges of its influence to prevent discontinuities. In principle, the weights can be computed to optimize the quality of the EigenSkin correction, and this is a topic of future research.

2.2 Locally Supported Joint Displacements

Let $\mathscr{P}$ be the set of indices of observed poses with $0 \in \mathscr{P}$ representing the rest pose and let the observed vertex positions and bone transforms for pose $p \in \mathscr{P}$ be denoted as v^p and T^p, respectively. The differences between the SSD vertex positions and the observed pose positions mapped back into the rest pose yield displacements (see Figure 2),

$$u_i^p = \left(\sum_{b\in B_i} w_{ib} T_b^p\right)^{-1} v_i^p \;-\; v_i^0.$$

The observed mesh shapes result when these displacements are added to the rest pose before applying the bone weighted transformation. If the deformations vary smoothly over pose space, then interpolated displacements provide a good approximation of deformations at configurations between observations.

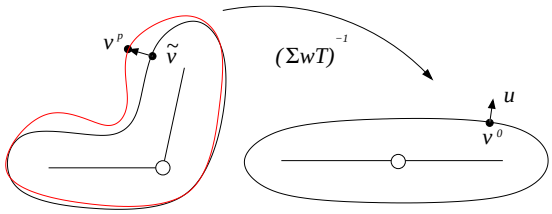


Figure 2: The displacement for vertex i and pose p, denoted u_i^p, is the difference between its observed position, v_i^p, and its position under SSD, $\tilde{v}_i^p$, mapped back into the rest pose.

To make our hardware implementation possible, we exploit the observation that localized changes to the configuration of an articulated character often result in local deformations. This independence occurs in most articulated characters, and certainly exists in realistic human hands. Bending a single joint in one finger, though difficult without bending any other joints, does not cause noticeable deformations in the other fingers. Likewise, bending one finger of our finite element hand model does not cause noticeable deformations in the others (see Figure 4). Although the finite element model deformations resulting from a change to a single joint are global, the displacement magnitudes are imperceptible at vertices that are far from the joint. We refer to the set of vertices significantly affected by a joint motion as the *joint support*. Note that the joint supports depend on the SSD weights and in general they do not correspond to the sets of vertices influenced by bone transforms.

To find the support of a joint we compute the deformations that result from moving the joint to different positions in its full range of motion while keeping all other joints fixed to the rest pose position. The set of vertices having a displacement larger than a given threshold in any of these computed poses then becomes the support of this joint. For example, in our case we used four percent of the maximum observed displacement (we will see that memory constraints also play a large part). Several joint supports of our finite element hand model are shown in Figure 3. Note that we consider only single joint perturbations due to the high dimensionality of our hand model's configuration space. Nevertheless, we can still approximate linear coupling effects since we let the joint supports overlap.

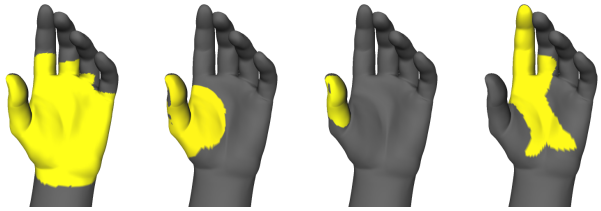


Figure 3: *Joint supports* for thumb carpal-metacarpal, metacarpo-phalangeal, inter-phalangeal joints and index metacarpo-phalangeal joint.

For notational convenience, suppose the articulated figure has a tree structure, i.e., does not have loops, such as for humanoids, and joints are denoted by the index of the adjacent bone furthest from the root of the hierarchy. Denoting $0 \in \mathscr{B}$ as the root, joints have nonzero index. Let $P_j \subset \mathscr{P}$ be the set of pose indices used to compute the support for joint j and let S_j be the set of vertex indices in the joint support. Furthermore, let J_i be the set of joints whose supports contain vertex i. That is, $J_i = \{j | i \in S_j\} \subset \mathscr{B}\backslash\{0\}$.

2.3 Eigendisplacements

Although the pose displacements computed for independently perturbed joints may be used as a basis for describing displacements of new configurations, significant redundancy exists in the pose displacements, e.g., skin bulging in similar directions. Principal Component Analysis (PCA) of joint support displacements yields an orthogonal displacement basis, which we term *eigendisplacements*. As guaranteed by PCA, adding successive corrections with the eigendisplacement basis provides approximations which are better in a formal, least squares, sense [Golub and van Loan 1996].

Computing principal components with the Euclidean norm is equivalent to computing the singular value decomposition (in the case of a square symmetric matrix it is equivalent to eigenanalysis). For each joint j we construct a rectangular matrix, $\mathbf{A}_j$, of size $3|S_j| \times |P_j|$, whose columns consist of the x,y, and z components of the vertex displacements on the joint support. In the singular value decomposition, $\mathbf{A}_j = \mathbf{U}_j \mathbf{D}_j \mathbf{V}_j^T$, the matrix $\mathbf{U}_j$ has the same size as $\mathbf{A}_j$ and consists of columns of eigendisplacements for support j in the same block column format that was used to build $\mathbf{A}_j$. The singular values, in the diagonal matrix $\mathbf{D}_j$, identify the importance that each eigendisplacement has in reproducing the observed poses (they relate to the proportion of variation explained by each principal component). Note that the matrix $\mathbf{V}_j$ and the singular values combine to gives the coordinates of our observed displacements in

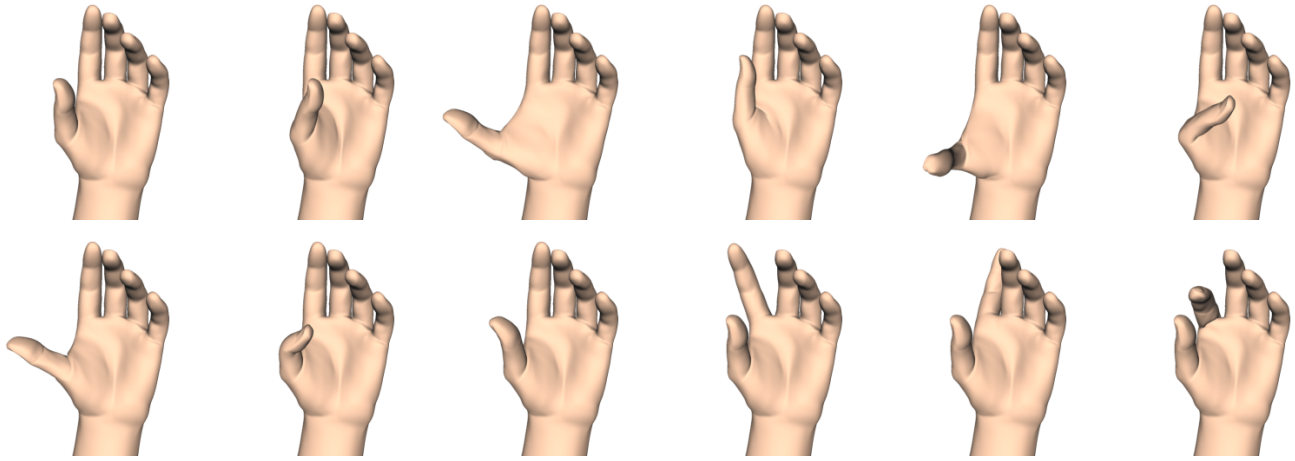


Figure 4: A subset of the training data showing some of the thumb and index finger poses.

the eigendisplacement basis. We denote $\hat{u}_i^{jk}$ the eigendisplacement of vertex i in the basis of support j with importance k where k goes from 1 (the principal component) up to $|P_j|$. Figure 5 shows the first four eigendisplacements of the thumb carpal-metacarpal joint support in our hand example.

At this point we can truncate each eigendisplacement basis expansion knowing that the error will be minimized in the least squares sense. The hardware limits the size of each truncated basis set as there is a limited amount of per vertex data memory in which we can send the eigendisplacements to the EigenSkin vertex program (see Section 2.5). Letting $n_j < |P_j|$ be the size of the truncated basis set of joint support j, this constraint can be written as

$$\max_i \; n_j |J_i| \;\leq\; \text{maximum possible displacements.}$$

Instead of choosing each n_j individually, we take an equal number of eigendisplacements from each support.

The equation for computing the deformed mesh shape for an arbitrary configuration with bone transforms $\{T_b\}_{b\in\mathscr{B}}$ can then be written as

$$\tilde{v}_i = \sum_{b\in B_i} w_{ib} T_b \left(v_i^0 + \sum_{j\in J_i} \sum_{k=1}^{n_j} \tilde{\alpha}_{jk} \hat{u}_i^{jk} \right) \tag{2}$$

where $\tilde{\alpha}_{jk}$ gives the *coordinates* of the displacement correction in terms of the reduced eigendisplacement basis. These coordinates are computed to interpolate between observed displacements, as shown below. Note that Equation 2 provides a powerful model for shape deformation (see, in particular, [James and Pai 2002a]).

2.4 Interpolating Eigendisplacement Coordinates

As an articulated character moves between observed configurations, its shape should interpolate the observed poses. To do this we interpolate the eigendisplacement coordinates of the observed configurations. For the truncated set of eigendisplacements at each support, we need the coordinates in the truncated basis which give displacements closest to the observed displacements. That is, we want to solve for α^p in

$$u_i^p = \sum_{j\in J_i} \sum_{k=1}^{n_j} \alpha_{jk}^p \hat{u}_i^{jk}.$$

This is an over constrained linear system which we can solve using least squares to get the best fit to our observed displacements. Conveniently, the least squares solution for any number of eigendisplacements, n_j, is available from the singular value decomposition computed in Section 2.3. For joint support j, column p of $\mathbf{D}_j\mathbf{V}_j^T$ contains α_{jk}^p for $k = 1..|P_j|$.

This leads us to the problem of computing the eigendisplacement coordinates for arbitrary configurations. Radial basis functions [Powell 1987] (RBF) are a common choice for interpolating scattered data, and have been used by Lewis et al. [Lewis et al. 2000] for pose space deformation and by Sloan et al. [Sloan et al. 2001] for shape interpolation with articulated figures. Our interpolation is one dimensional since all our observations involved perturbations of individual joints. Although we could use a simpler interpolant, we also choose RBFs because they extend easily to the higher dimensional domains needed to let EigenSkin capture nonlinear multi-joint coupling effects (a subject of future work).

We use Gaussian interpolation shape functions, $\phi(r) = \exp(-r/r_0)$. In our one dimensional case, the α_{jk} only depend on the distance of joint j from its settings in poses P_j. For revolute joints, we can easily compute the distance, r, by comparing the joint angles directly. For joints with more than one rotational degree of freedom, we compute distance as the angle in the axis-angle representation of the joint's rotation matrix.

Ideally, with a large number of observed joint perturbations per support we would interpolate using fewer interpolation basis functions (ϕ) than observations. In the case of our hand model, however, we only have approximately half a dozen pose perturbations for each joint degree of freedom (for a total of approximately 120 poses). This justifies our use of interpolation basis functions since the total cost of constructing and evaluating the RBF interpolant for half a dozen poses is negligible. The interpolated eigendisplacement coordinates for a new pose are computed as

$$\tilde{\alpha}_{jk} = \sum_{q\in P_j} \lambda_q^{jk} \phi(r_{jq})$$

where r_{jq} is the distance of joint j in the new pose from its setting in pose q, and the λ_q^{jk} for $q \in P_j$ are given by the solution to the linear system,

$$\alpha_{jk}^p = \sum_{q\in P_j} \lambda_q^{jk} \phi(r_{jq}^p), \;\text{ for }\; p \in P_j.$$

Here r_{jq}^p is the distance between joint j's position in pose p and its position in pose q (and thus $r_{jp}^p = 0$). The system of equations

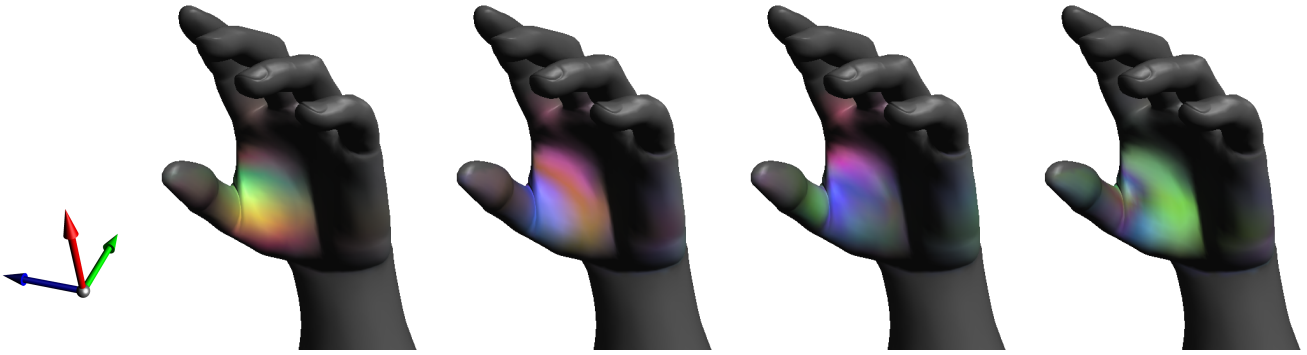


Figure 5: *Eigendisplacements* and singular values, σ, for thumb carpal-metacarpal joint in left-right order of importance. XYZ components of displacement are represented using an RGB *colour* correspondence.

is square, and invertible provided P_j does not contain two observations with identical joint settings.

2.5 EigenSkin Vertex Programming

Modern vertex programming hardware (e.g., [Lindholm et al. 2001]) is ideally suited to performing the per-vertex weighted linear superposition of eigendisplacements (contained in the large brackets of Equation 2) performed prior to the SSD weighted transformation. Depending on the number of eigendisplacements used, the weighted eigendisplacement vector accumulations are about as costly as the weighted transform matrix-vector multiply-accumulate operations.

Current vertex programs limit per vertex data to 16 4-tuples of floats. In our implementation we impose a limit of 10 eigendisplacements per vertex (or 5 eigendisplacements and 5 normal corrections), which still leaves room for texture coordinates after specifying the vertex position, normal, colour, and bone weights. Notice that this limit is not hard since careful choices and packing of per vertex data permit more than 10 of the 16 available tuples to be allocated for EigenSkin data.

If a vertex is in many supports then the number of eigendisplacements renderable by current hardware may be too severely restricted. In this case it is useful to smoothly mask the support groups to smaller regions, otherwise fewer eigendisplacements must be used.

3 Results

To illustrate our EigenSkin method, we have constructed a finite element model of the human hand (see Figure 6) which exhibits subtle nonlinear skin deformations. The surface skin model and matching skeleton are based on Loop subdivision [Loop 1987] of a hand mesh exported from Curious Labs Poser [Curious Labs Inc.]. A finite element mesh containing 11,171 high-order 10-node tetrahedral elements was generated using NETGEN [Schoberl 1997] (and subsequent simplification). The hand was moved into various poses by applying position constraints to vertices adjacent to the rigid bones, and computing the resulting tissue deformation using geometrically nonlinear static finite element analyses [Zienkiewicz 1977] with (a modified version of) the CalculiX program [Dhondt and Wittig]. Approximately half a dozen poses were computed for each joint degree of freedom to estimate the locally supported joint eigendisplacements, and 25 additional poses were computed for validation. Finite element analyses were performed on a cluster of modern workstations and consumed several hundred CPU hours. The model was not intended to reproduce detailed skin wrinkling effects, and lacks anatomical details such as tendons, blood vessels, and skin layers. Despite these limitations, the model reasonably describes bulk tissue deformations and was sufficient to illustrate our method.

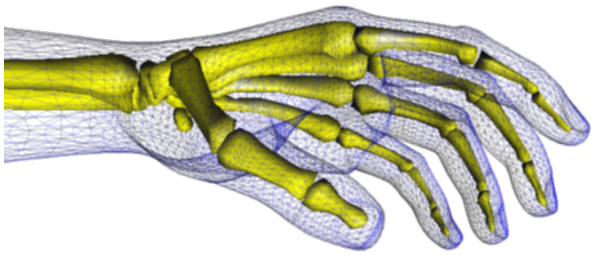


Figure 6: Skeleton used to drive finite element hand model.

As shown in Figure 7, the eigendisplacement approximations of the hand model produce a clear improvement over the traditional SSD algorithm. Even with only five leading eigendisplacements, the EigenSkin approximation is essentially indistinguishable from the original FEM model.

Our interactive simulation uses a CyberGlove [Immersion Corporation] input device to interactively drive our EigenSkin hand model, while graphical feedback is rendered using OpenGL and a GeForce3 graphics card. Radial basis function interpolation of the pose-space data is performed on the main CPU, with eigendisplacement amplitudes and bone transforms set as input parameters to the EigenSkin vertex programs which are compiled as static display lists. Currently, our unoptimized implementation renders the EigenSkinned hand model only slightly slower than the traditional SSD model. A large 55,904 triangle hand model renders at 47 frames per second (FPS), while a coarser 13,976 triangle model achieves 181 FPS. Please see our accompanying video for a demonstration of the real time simulation.

4 Conclusions and Discussion

Our results confirm that the *EigenSkin* method is an effective tool for character skinning when compressed hardware renderable approximations are required for an articulated character's nonlinear quasi-static deformations. *EigenSkin* works best when SSD corrections are localized, providing independence between different parts of the mesh, and are stable (i.e., corrections vary slowly over pose-space), allowing accurate and efficient interpolation. Under these

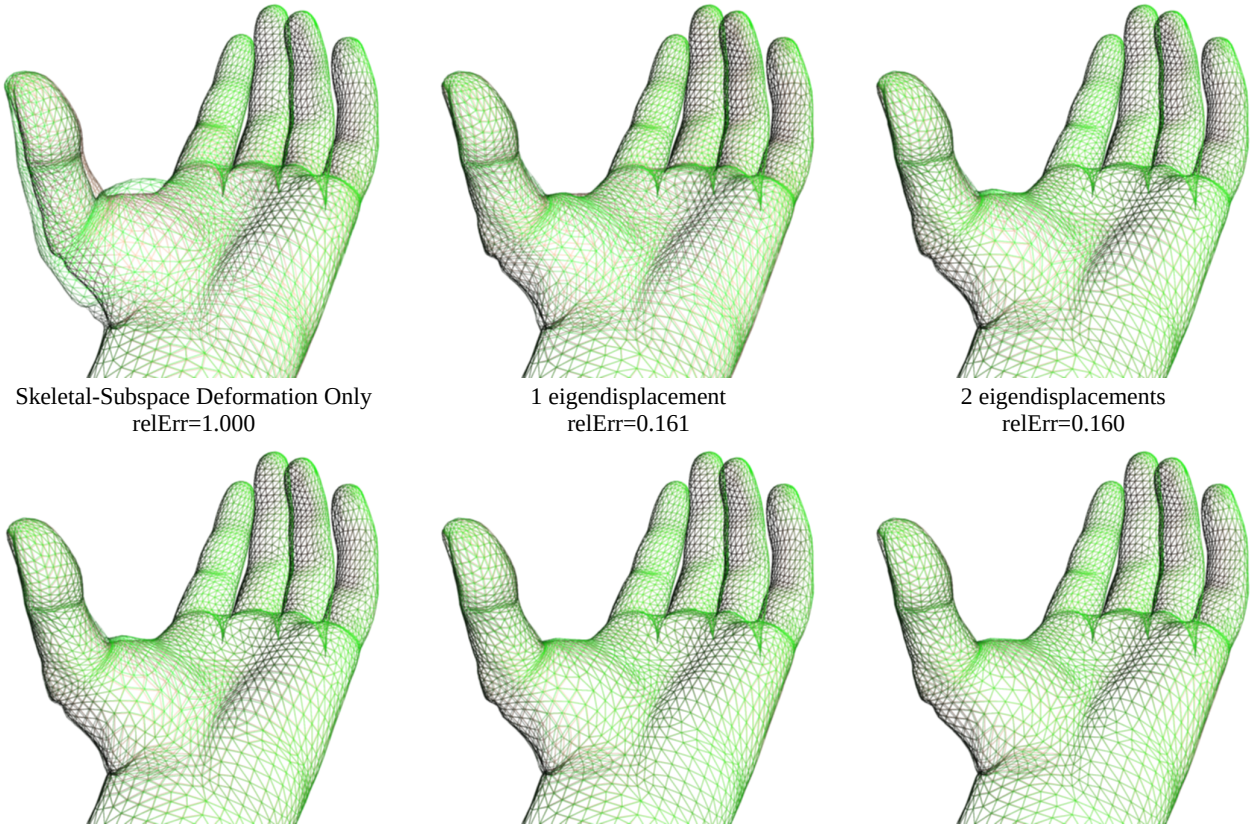


Figure 7: *EigenSkin approximation* for 0–5 eigendisplacements of the thumb's carpal-metacarpal joint shown with the 13,976 triangle hand model. The pose geometry (skin colour) is approximated by the EigenSkin model (green). The l_2 relative displacement error (relErr) is also printed below each image. Remarkably, *the SSD model is substantially improved after the addition of only one eigendisplacement.*

conditions, very practical results can be obtained in which only one or two eigendisplacements per joint produce a visually dramatic improvement over commonplace Skeletal-Subspace Deformation.

Limitations and Future Work: Despite the advantages of EigenSkin, as presented there are several limitations to the method which are topics of current research. We assume that an initial SSD model is provided and then show how the EigenSkin corrections are beneficial. However, an alternate approach involves optimizing bone weights to allow better EigenSkin approximations of the displacements and normals. While good eigendisplacement bases can often be constructed using displacements resulting from single joint motions, in practice it is desirable to allow general pose sets and to recover nonlinear joint-joint coupling phenomena.

References

ALLEN, B., CURLESS, B., AND POPOVIC, Z. 2002. Articulated body deformation from range scan data. In *SIGGRAPH 02 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3D faces. In *SIGGRAPH 99 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

BOOKSTEIN, F. L. 1989. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 6 (June).

CANI-GASCUEL, M.-P. 1998. Layered Deformable Models with Implicit Surfaces. In *Graphics Interface*, 201–208.

CHEN, D. T., AND ZELTZER, D. 1992. Pump it up: Computer animation based model of muscle using the finite element method. In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, Addison Wesley, vol. 26, 89–98.

COTIN, S., DELINGETTE, H., AND AYACHE, N. 1999. Realtime Elastic Deformations of Soft Tissues for Surgery Simulation. *IEEE Transactions On Visualization and Computer Graphics 5*, 1, 62–73.

CURIOUS LABS INC. Poser 4, Santa Cruz, CA., http://www.curiouslabs.com/products/poser4.

DEBUNNE, G., DESBRUN, M., BARR, A., AND CANI, M.-P. 2001. Dynamic Real-Time Deformations Using Space and Time Adaptive Sampling. In *SIGGRAPH 01 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

DHONDT, G., AND WITTIG, K. CalculiX: A Free Software Three-Dimensional Structural Finite Element Program, http://www.calculix.de.

GOLUB, G. H., AND VAN LOAN, C. F. 1996. *Matrix Computations*, third ed. Johns Hopkins University Press, Baltimore and London.

GOURRET, J., MAGNENAT-THALMANN, N., AND THALMANN, D. 1989. Simulation of Object and Human Skin Deformations in a Grasping Task. In *Computer Graphics (SIGGRAPH 89 Conference Proceedings)*, Addison Wesley.

HERMAN, D. L. 2001. Using Precomputed Cloth Simulations for Interactive Applications. In *SIGGRAPH 2001 Conference Abstracts and Applications*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

HYVARINEN, A., OJA, E., AND KARHUNEN, J. 2001. *Independent Component Analysis*. Wiley, Johns and Sons, Inc.

IMMERSION CORPORATION. CyberGlove, http://www.immersion.com.

JAMES, D. L., AND PAI, D. K. 1999. ARTDEFO: Accurate Real Time Deformable Objects. In *SIGGRAPH 99 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH, 65–72.

JAMES, D. L., AND PAI, D. K. 2002. DyRT: Dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH 02 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

JAMES, D. L., AND PAI, D. K. 2002. Real time simulation of elastokinematic models. In *ICRA2002: IEEE International Conference on Robotics and Automation*.

JOLLIFFE, I. 1986. *Principal Component Analysis*. Springer Verlag.

LEE, Y., TERZOPOULOS, D., AND WALTERS, K. 1995. Realistic Modeling for Facial Animation. In *SIGGRAPH 95 Conference Proceedings*, Addison Wesley, vol. 29 of *Annual Conference Series*, ACM SIGGRAPH, 55–62.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose Space Deformations: A Unified Approach to Shape Interpolation a nd Skeleton-Driven Deformation. In *SIGGRAPH 00 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

LINDHOLM, E., J.KILGARD, M., AND MORETON, H. 2001. A User-Programmable Vertex Engine. In *SIGGRAPH 01 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

LOOP, C. 1987. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, University of Utah, Department of Mathematics.

MAESTRI, G. 1999. *Digital Character Animation 2, Vol. 1*. New Rider, Indianapolis.

MAGNENAT-THALMANN, N., AND THALMANN, D. 1991. Human Body Deformations Using Joint-dependent Local Operators and Finite Element Theory. In *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*. Morgan Kaufmann, 243–262.

MAGNENAT-THALMANN, N., LAPERRIERE, R., AND THALMANN, D. 1988. Joint-dependent Local Deformations for Hand Animation and Object Grasping. In *Proc. of Graphics Interface '88*, 26–33.

METAXAS, D., AND TERZOPOULOS, D. 1992. Dynamic Deformation of Solid Primitives with Constraints. In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, Addison Wesley, vol. 26, 309–312.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical Modeling and Animation of Brittle Fracture. In *SIGGRAPH 99 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH, 111–120.

PAI, D. K., VAN DEN DOEL, K., JAMES, D. L., LANG, J., LLOYD, J. E., RICHMOND, J. L., AND YAU, S. H. 2001. Scanning Physical Interaction Behavior of 3D Objects. In *SIGGRAPH 01 Conference Proceedings*, Addison Wesley, Annual Conference Series, ACM SIGGRAPH.

PARKE, F. I., WATERS, K., AND PARKE, F. I. 1996. *Computer Facial Animation*. A. K. Peters Ltd.

PENTLAND, A., AND WILLIAMS, J. July 1989. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (SIGGRAPH 89 Conference Proceedings) 23*, 3, 215–222. Held in Boston, Massachusetts.

PICINBONO, G., DELINGETTE, H., AND AYACHE, N. 2001. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference on Robotics and Automation*.

POWELL, M. J. D. 1987. Radial basis functions for multivariate interpolation: A review. In *Algorithms for Approximation*, J. C. Mason and M. G. Cox, Eds. Clarendon Press, Oxford.

SCHEEPERS, F., PARENT, R. E., CARLSON, W. E., AND MAY, S. F. 1997. Anatomy-Based Modeling of the Human Musculature. In *SIGGRAPH 97 Conference Proceedings*, Addison Wesley, vol. 31 of *Annual Conference Series*, ACM SIGGRAPH, 163–172.

SCHOBERL, J. 1997. NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules. *Comput.Visual.Sci 1*, 41–52.

SINGH, K., AND KOKKEVIS, E. 2000. Skinning Characters using Surface-Oriented Free-Form Deformations. In *Graphics Interface 2000*, 35–42.

SLOAN, P.-P. J., ROSE, C. F., AND COHEN, M. F. 2001. Shape by example. In *2001 Symposium on Interactive 3D Graphics*, 135–143.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Deformable Models. *The Visual Computer 4*, 306–331.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically-based Models with Rigid and Deformable Components. *IEEE Computer Graphics and Applications 8*, 6, 41–51.

TURK, M., AND PENTLAND, A. 1991. Eigen faces for recognition. *Jrnl. Cognitive Neuroscience 3*, 71–86.

WILHELMS, J., AND VAN GELDER, A. 1997. Anatomically Based Modeling. In *SIGGRAPH 97 Conference Proceedings*, Addison Wesley, vol. 31 of *Annual Conference Series*, ACM SIGGRAPH, 173–180.

ZHUANG, Y., AND CANNY, J. 1999. Real-time Simulation of Physically Realistic Global Deformation. In *IEEE Vis'99 Late Breaking Hot Topics*.

ZIENKIEWICZ, O. C. 1977. *The Finite Element Method*. McGraw-Hill Book Company (UK) Limited, Maidenhead, Berkshire, England.