Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

Controllable Hand Deformation from Sparse Examples with Rich Details

Haoda Huang[†] Ling Zhao[*] KangKang Yin[‡] Yue Qi[*] Yizhou Yu[§] Xin Tong[†]

[†] Microsoft Research Asia [*] Beihang University [‡] National University of Singapore [§] Univ. of Illinois at Urbana-Champaign

Abstract

Recent advances in laser scanning technology have made it possible to faithfully scan a real object with tiny geometric details, such as pores and wrinkles. However, a faithful digital model should not only capture static details of the real counterpart but also be able to reproduce the deformed versions of such details. In this paper, we develop a data-driven model that has two components respectively accommodating smooth large-scale deformations and high-resolution deformable details. Large-scale deformations are based on a nonlinear mapping between sparse control points and bone transformations. A global mapping, however, would fail to synthesize realistic geometries from sparse examples, for highly-deformable models with a large range of motion. The key is to train a collection of mappings defined over regions locally in both the geometry and the pose space. Deformable fine-scale details are generated from a second nonlinear mapping between the control points and per-vertex displacements. We apply our modeling scheme to scanned human hand models. Experiments show that our deformation models, learned from extremely sparse training data, are effective and robust in synthesizing highly-deformable models with rich fine features, for keyframe animation as well as performance-driven animation. We also compare our results with those obtained by alternative techniques.

1. Introduction

Technology for laser range scanning has been significantly improved over the last decade in terms of both precision and speed. It has become possible to faithfully scan a real object with tiny geometric details, such as pores and wrinkles. However, many real objects including most natural organisms deform. A faithful digital model should not only capture static details of the real counterpart but also reproduce the deformed versions of such details. Data-driven methods are well-suited for this purpose for two reasons. First, it would be extremely expensive to physically simulate deformations of such high-resolution details. Second, fine-scale deformations of different organisms follow different styles. A data-driven method is more suitable to incorporate the unique characteristics of different types of deformation.

There exist two major challenges in building high-resolution data-driven deformation models. First, typically there is only a limited amount of training data available because of the huge amount of time and effort required to scan high-resolution details. Training data-driven models with sparse examples can easily result in inaccurate and misfit models that produce poor predictions. Second, from an animation perspective, we would like to generate realistic deformations from a sparse set of markers. This calls for a data-driven model that can correlate low-dimensional control signals with high-dimensional deformation details.

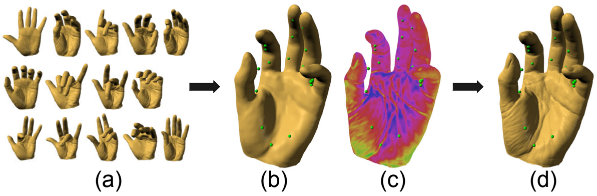


Figure 1: *Given sparse training examples (a), we train a collection of deformation models at two layers. Given a new configuration of the control points, these models can generate smooth large-scale deformations (b) and high resolution displacements (c), which are then combined to produce deformed models with rich details (d).*

[†] e-mail:{hahuang,xtong}@microsoft.com
[‡] e-mail:kkyin@comp.nus.edu.sg
[§] e-mail:yyz@illinois.edu

We propose a robust deformation framework to tackle the above challenges. Our data-driven model has two components accommodating smooth large-scale deformations and high-resolution deformable details, respectively. Large-scale deformations are based on linear blend skinning and nonlinear mappings between sparse control points and bone transformations. To alleviate mis-fitting caused by training with sparse nonlinear data, we train a collection of local mappings defined over the manifold of seen example poses. A 'pose' in this paper refers to a specific shape of a surface model or a specific configuration of its control points. Each of the local mappings takes one of the training poses as its reference pose, and its nearby poses as training inputs. The deformation associated with a novel pose is then predicted using a weighted mixture of local mappings defined for the poses closest to the target pose. This training process may, however, mistakenly learn false coupling among distant control points from sparse training examples even when constrained within local pose spaces. We address this problem by learning correlations only from control points and deformation regions geometrically located nearby.

High-resolution deformable details are modeled in a separate training pass, which learns a per-vertex nonlinear mapping between control points and per-vertex displacements. Several choices exist in terms of input signals for displacement modeling. We have confirmed that a direct mapping between the control points and the displacements is more effective than a cascaded mapping where large-scale deformation predictions are used to drive the deformations of details. We prefer 1D displacements along vertex normals to 3D displacement vectors to reduce nonlinearity and memory requirements.

The proposed modeling scheme is tested with scanned human hand models. To the best of our knowledge, few previous works test on human hands, which have large degrees of freedom, large ranges of motion, and highly deformable wrinkles. Our framework is capable of synthesizing high-resolution hand mesh animations with rich and varying details, from as few as 14 training examples. Our choice of control signals for the deformation models results in an easy-to-use keyframe animation system, as well as a promising way for performance-driven mesh animation.

2. Related Work

Data-driven Mesh Skinning For real-time applications, Linear Blend Skinning (LBS) is most widely used by artists because of its simplicity and efficiency. However, the original LBS suffers from 'candy-wrapper' artifacts. Pose Space Deformation (PSD) improves skinning quality by integrating LBS and RBF-based interpolation [LCF00]. More advanced example-based techniques, such as [SRC01, MG03, WSLG07, WPP07], have been effectively integrated with mesh deformation algorithms to further improve the quality of skinning. EigenSkin models the residual errors of LBS using principal component analysis [KJP02]. [JT05] extracts proxy bones and their influence weights automatically from animated mesh sequences. We use the same method to extract virtual bones from our example surface models. In [PH06], high quality skin deformations are reconstructed via a second-order skinning scheme followed by RBF-based interpolation of the residual errors. DrivenShape [KV08], an interesting technique that exploits known correspondences between two sets of deformation examples, further pushes the line where data-driven shape deformations can be applied. These methods, however, do not support direct manipulations or handle fine-scale features at the wrinkle level. Most of them require dense example data as well.

The work of [KM04, SZGP05, DSP06, FKY08] are the closest in spirit to our own. Kurihara and Miyata [KM04] use a per-vertex weighting scheme for PSD to animate hand meshes from sparse examples. They can handle large-scale deformations well but not the details, and they do not support direct manipulation with low-dimensional control signals. Mesh-based Inverse Kinematics (MeshIK) adopts a *global* weighting scheme where all vertices from the same example mesh are given the same weight [SZGP05, DSP06]. Such global scheme produces artifacts when modeling from sparse examples, for both large-scale and fine-scale deformations. In the absence of a skeleton, Feng et al. [FKY08] build a *global* data-driven mapping between sparse control points and proxy bone transformations for predicting novel surface deformations in real time. There are two limitations with this method. First, prediction errors may increase significantly when a novel control point configuration deviates far away from the reference configuration. Second, given sparse training examples, false dependencies between distant object parts may be mistakenly enforced by the global mapping. Our method addresses these challenges by learning local deformation models in both the geometry and the pose space. We compare our results with respect to those obtained from MeshIK and [FKY08] in Section 7.

Detail Modeling for Mesh Animation Detail modeling for mesh animation has been attracting more and more research effort in recent years, mainly due to the advances in acquisition techniques. Several multi-scale deformation schemes have been proposed for face modeling [BBA*07, BLB*08, MJC*08], and they all represent large-scale deformations with thin shell models. Fine-scale geometric details such as wrinkles are modeled using 2D splines [BBA*07], pose-space interpolations [BLB*08], or polynomial displacement maps [MJC*08]. We target highly-deformable models with larger ranges of motion and deformation, such as human hands, for which techniques developed for 2.5D surfaces such as faces cannot directly be applied. Furthermore, these methods require dense markers and training data for motion tracking and deformation modeling, while our method only needs a sparse set of control points and a sparse set of training examples. More recently, body parts or full-body geometries can be reconstructed from single-view or multiple-view dense video sequences [dAST*08, VBMP08, LAGP09]. The reconstructed geometries usually lack details, and do not generalize beyond seen examples.

Data Acquisition Generally speaking, acquiring high-resolution 3D models with fine features is difficult, expensive, and time-consuming. Structured light and photometric stereo are commonly used to capture 3D geometries in-house, especially for facial expressions. The quality of the models depends on the equipment and reconstruction algorithms used. A template-based method is employed in [ZSCS04] to produce point correspondences across an entire video sequence without using any markers, while [MJC*08] uses 178 markers for registration. [GMP*06] captures static faces with a commercial face-scanning system to model aging effects. Deformations, large or small, are not considered there. [PH06] uses a commercial motion capture system and 350 markers to capture medium-scale muscle deformations for full-body motions, but fine-scale skin movements are hard to capture using motion capture systems alone. We use a high-precision commercial 3D scanner to capture hand models. Although our deformation technique is independent of the underlying geometry acquisition method, and can animate models captured by other means.

3. Overview

Our system consists of an offline training stage, and an online synthesis stage, as shown in Figure 2. The training examples, P_i, are high-resolution meshes with rich details. They are first registered with respect to each other, in terms of both large-scale and fine-scale features. The model registration process will be described in §4. We denote the output of model registration as $\widetilde{P}_i$, which are a collection of low-resolution smooth meshes of the same topology. We decompose the deformation learning into two layers: bone-based transformation modeling for large-scale deformations, and displacement modeling for fine-scale details. The low-resolution $\widetilde{P}_i$ are used to train the large-scale deformation models. To train the fine-scale detail models, we extract the differences between P_i and $\widetilde{P}_i$ as displacement maps $\mathbf{h}_i$, which capture high frequency deformation details.

Both deformation layers learn regression models with the same set of control knobs, i.e., the control points $\mathbf{c}$. A global regression extracts correlations between the full control point vector $\mathbf{c}$ and transformations $\mathbf{d}_j$ of bone b_j from all example poses, and generates a single prediction model $\mathbf{d}_j(\mathbf{c})$ for each bone. In contrast, we train a collection of models $\mathbf{d}^i_j(\mathbf{c}_j)$, where $i = 1, \ldots, \#poses$, $j = 1, \ldots, \#bones$. These models are local in both the geometry space and the pose space. The learning methods will be given in §5.1, together with necessary implementation details. Building local deformation models both in the geometry and pose space effectively eliminate false coupling of independent object parts and severe model mismatch for nonlinear sparse training examples.

Displacement maps $\mathbf{h}_i$ are modeled in another pass of regression as $\mathbf{h}(\mathbf{c})$. We will detail this process in §5.2. This pass of vertex-level displacement regression is to model the myriads of variations of fine details, such as wrinkles and palm lines, which are beyond the modeling capability of linear blend skinning.

At runtime, new control point configurations drive the learned models $\mathbf{d}^i_j(\mathbf{c}_j)$ and $\mathbf{h}(\mathbf{c})$ to produce new poses with plausible large-scale deformations as well as realistic variations of fine features. §6 describes the necessary formulas for deformation synthesis.

4. Model Registration and Detail Extraction

Our training meshes are independently scanned, so initially they reside in different coordinate systems. We therefore rotate and translate the training examples $P_i, i = 0...n-1$ into the coordinate frame of a chosen reference mesh P_0. More specifically, we interactively identify a small number of corresponding points between each training mesh and P_0 to resolve the rigid transformations between them, so that the differences among the rigidly transformed meshes are the deformations we wish to model.

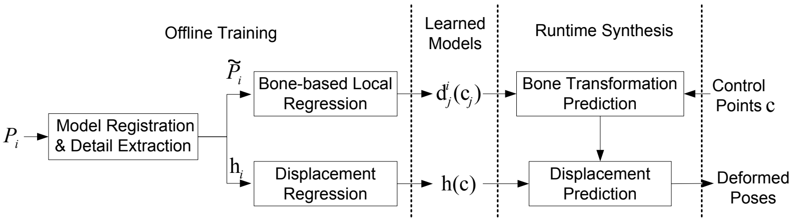
We further employ deformation transfer [SP04] to build per-vertex correspondences among the training models. Note that deformation transfer enforces a smoothness constraint on nearby vertices, and is thus not suitable for meshes with high-frequency details. Therefore we apply mesh retiling and Laplacian smoothing techniques [Tur92] to obtain a collection of smooth meshes, $\overline{P_i}$, at a lower resolution. The smooth reference model $\overline{P_0}$ is then deformed towards each training model $\overline{P_i}$, and we denote the deformed reference models as $\widetilde{P}_i$. $\widetilde{P}_i$ will be used to model large-scale deformations, each of which possesses the shape of $\overline{P_i}$, but the topology of $\overline{P_0}$. A homogeneous mesh topology can also facilitate fine-scale feature extraction for detail modeling.

To extract the differences between the original high-resolution meshes P_i and their smoothed low-resolution version $\widetilde{P}_i$, we subdivide $\widetilde{P}_i$ to retrieve the resolution of the original mesh P_i. At each vertex of the subdivided $\widetilde{P}_i$, a per-vertex displacement with respect to P_i is calculated along the vertex normals. We denote these displacement maps $\mathbf{h}_i$, and they will be used to train deformation models for fine surface features.

5. Deformation Modeling

We first present the modeling component of our deformation framework, which consists of two layers: large-scale bone-based deformation models, and fine-scale vertex-based displacement models.

Figure 2: *The training and runtime phases of our deformation framework. P_i: high-resolution training examples; $\widetilde{P}_i$: low-resolution registered meshes; $\mathbf{h}_i$: displacement maps;* **c***: control points;* $\mathbf{d}^i_j(\mathbf{c}_j)$*: bone-based large-scale deformation models;* $\mathbf{h}(\mathbf{c})$*: vertex-level detail models.*



5.1. Large-scale Deformation Modeling

Large-scale low frequency deformations are usually generated from bone and muscle motions. We therefore follow the conventional bone-based linear blend skinning to generate large-scale deformations. From the registered input models $\widetilde{P}_i$, we obtain transformations of individual triangles. We then cluster triangles of similar transformations to form abstract bones [JT05, FKY08], as shown in Figure 3(a). Note that the virtual bones generally do not conform to the biological bones anatomically, for instance more than a thousand bones are generated for the hand models. Each bone acts as an abstract representative for transformations, and its influence weights on a vertex are then obtained by minimizing the fitting error of vertex positions in all examples. Note that the large number of abstract bones is to guarantee the accuracy of the deformation models. Since they are automatically generated, no extra work is required from the users.

Denote the fitted influence bone set for vertex v as $B(v)$, and its skinning weight from bone b_j in $B(v)$ as w_j. The skinned vertex position v is computed using a weighted average of transformations from its influence bones:

$$v = \sum_{j \in B(v)} w_j \mathbf{T}_j v^r \qquad (1)$$

where $\mathbf{T}_j$ is the transformation matrix of bone b_j, and v^r represents the vertex position in the reference pose.

The task of the large-scale deformation modeling is to learn models of the form $\mathbf{T}_j(\mathbf{c})$ to predict bone transformations from control points **c**. We choose to use a quaternion representation for bone rotations $\mathbf{d}_j$ only, and solve for bone translations using a Poisson solver. Predicted bone rotations and solved bone translations together can then be converted to $\mathbf{T}_j$ appropriately for skinning.

As mentioned in Sections 1 and 3, learning large-scale deformation models globally from nonlinear sparse training data suffers from false correlation and mis-fitting. Figure 7(a) previews some artifacts resulted from such poor global models. In the pursuit of a robust system for deformation modeling and synthesis, we found that local learning in both the geometry space and the pose space is crucial.

5.1.1. Local regression in geometry space

The Degrees of Freedom (DoFs) of our hand models are extremely large. There are at least 21 DoFs associated with the skeletal structure of a hand, five for the thumb and four for each of the fingers. There are correlations between nearby structures, but faraway bones such as the thumb and the pinky can move relatively independently. Basic correlation analysis may, however, enforce unnecessary constraints on the movement of a bone with respect to that of a distant control point. This causes severe model mismatches when the training data is nonlinear and sparse. We therefore only train prediction models for bone rotations from spatially close control points, to decouple the accidental correlations between distant bones and control points seen from a few examples. We note that [LCXS10] also pointed out as a future direction to use region-based model learning to allow fine-grained control over local geometry and to improve the generalization ability of their models.

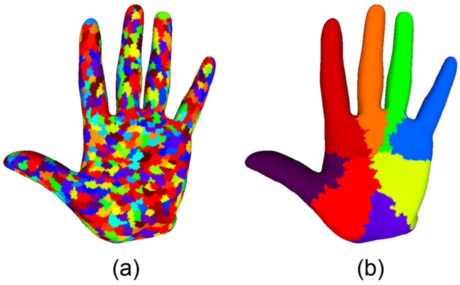


Figure 3: *Color-coded abstract bones (a) and regions sharing the same set of local control points (b).*

To locate the local control points for a particular bone b_j, the center location of all the vertices controlled by this bone is calculated, and its nearest vertex on the mesh is denoted as v_j. The k_c nearest control points to v_j, measured by the geodesic distance, are collected as the influence control point set $\mathbf{c}_j$ for b_j. Instead of learning deformation models from $\mathbf{d}_j$ and the full control point vector **c**, we now use $\mathbf{c}_j$ to learn a geometrically local deformation model of the form $\mathbf{d}_j(\mathbf{c}_j)$. We set $k_c = 7$ in all our experiments. Figure 3(b) visualizes bones sharing a same set of influence control points in the same color. This map conforms well with anatomical regions such as fingers.

5.1.2. Local regression in pose space

The range of motion of human hands, the most dexterous part of the human body, is enormous and highly nonlinear. Yet we only have 14 training examples in total. These ex-

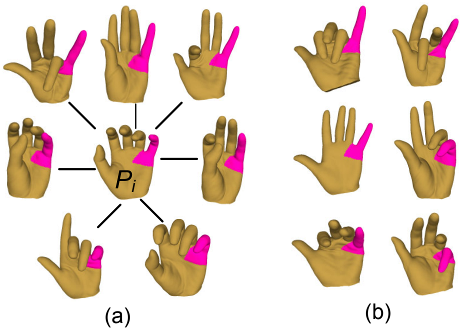


Figure 4: *The similarity graph for the pinky region (a). Distant poses in (b) are not used in training the local deformation models* $\mathbf{d}^i_j(\mathbf{c}_j)$.

amples appear extremely distant with respect to each other inside the huge configuration and deformation space of human hands. A global fit using all the example poses as so far described results in models with poor prediction results. Inspired by the success of local model learning and manifold learning methods, such as kNN (k-nearest neighbours), LWR (locally weighted regression) [CD88], and dimensionality reduction where local proximities are preserved rather than global proximities [SR03], we advocate building local regression models for each pose from their neighboring poses only.

To build local prediction models in the pose space, we first construct a weighted graph based on *local similarity* [BN01]. For a particular bone b_j, each example pose is connected with its k_p nearest neighbors ($k_p = 7$ in all our experiments), measured by the Euclidean distance of their influence control point vectors, with each edge weighted by a heat kernel as follows:

$$w(\mathbf{c}^i_j, \mathbf{c}^l_j) = e^{-|\mathbf{c}^i_j - \mathbf{c}^l_j|^2/2\sigma^2} \qquad (2)$$

where $\mathbf{c}^i_j$ represents the control point vector for bone b_j in example pose P_i. Figure 4 shows such a graph for the bones in the pinky region. To train the local model at pose $\widetilde{P}_i$ for bone b_j, we compute the relative rotation $\mathbf{d}^i_j$ of this bone between $\widetilde{P}_i$ and each of its neighboring pose in the similarity graph. $\mathbf{d}^i_j$ and $\mathbf{c}_j$ are then used to train a deformation predictor $\mathbf{d}^i_j(\mathbf{c}_j)$. These models, though unable to predict deformations for poses far away, are much more accurate locally than a global predictor. Again, eager readers can preview a comparison between global models and local models in Figure 7.

5.1.3. Implementation details

We perform linear CCA (Canonical Correlation Analysis) regressions to learn $\mathbf{d}^i_j(\mathbf{c}_j)$, and use the kernel trick to establish nonlinear dependencies between variables, similar to [FKY08]. That is, $\mathbf{d}^i_j(\mathbf{c}_j) = \mathbf{M}^i_j(\xi(\mathbf{c}_j))$, where $\mathbf{M}^i_j$ is a linear operator composed of several linear mappings, and $\xi(\mathbf{c}_j)$ is the kernelized vector of the input $\mathbf{c}_j$. We use Gaussian kernels for all our experiments.

Because of the extremely low number of training examples in our case, we only train bone rotation predictors, in the format of quaternions rather than eight-dimensional dual-quaternions [KCvO07], to reduce over-fitting. Bone translations are solved by the Poisson translation solver [FKY08], which minimizes a weighted sum of the edge prediction differences E_e and the control point positional errors E_c of the form $(\sum E_e + \beta \sum E_c)$. β is a weighting factor to control how exactly the surface should follow the control points. We use different weighting schemes for keyframe animation and performance-driven animation, which will be further described in the results section. The Poisson minimization equates to a linear least-squares problem whose solution can be written as $\mathbf{t} = \mathbf{Pf}$, where $\mathbf{t}$ is the vector of bone translations, $\mathbf{P}$ is a precomputed pseudo inverse matrix, and $\mathbf{f}$ contains both the bone rotations $\mathbf{d}$ and the control point positions $\mathbf{c}$. From $\mathbf{t}$ and $\mathbf{d}$ we can then easily compute the bone transformation matrices $\mathbf{T}$.

5.2. Fine-scale Displacement Modeling

What we have done in the previous sections is essentially linear blend skinning, even though the bone transformations are predicted from control point configurations that are local in both the geometry and the pose space. It is well known that linear blend skinning is ineffective in modeling high frequency deformation details. Using more abstract bones would improve the data fitting quality to some extent, but will run into the problem of over-fitting.

Therefore we train another layer of CCA-based regression models to account for the differences between P_i and $\widetilde{P}_i$. We use the high-resolution displacement maps $\mathbf{h}_i$ and their corresponding control points to train a displacement prediction model $\mathbf{h}(c)$ for every vertex. Another option we have tried is to use the predicted bone transformations $\mathbf{T}$ instead of the original control points $\mathbf{c}$ as input to the regression process. Our experiments show an inferior synthesis quality using such a cascaded scheme, as indicated by Figure 5(c), because the bone prediction errors are transferred to and amplified by the displacement predictor, leading to noticeable visual artifacts. Both Figure 5(b) and (c) show results after the Poisson reconstruction.

Different from the bone-driven large-scale deformation learning, it is not necessary to run the displacement regression locally either in the geometry space or in the pose space. The high frequency details are inherently local in nature, so distant regions and poses generally do not interfere with each other. In addition, we use 1D displacements along vertex normals rather than 3D displacement vectors. This not only improves model fit by factoring out nonlinearity in 3D rotations, but also reduces the storage consumption, because the displacement predictors are defined for each vertex of high-resolution models.

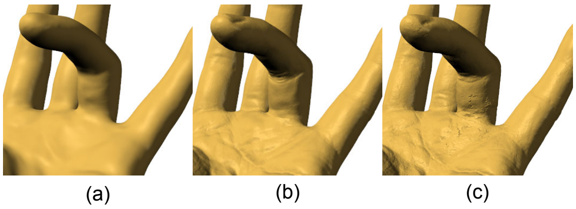


Figure 5: *Different strategies to model details. (a) The predicted smooth base mesh. (b) The base mesh plus details trained from control points. (c) The base mesh plus details trained from predicted bone transformations. Note the artifacts caused by error propagation.*

6. Deformation Synthesis

At runtime, the control points can either be manipulated by a user or driven by motion captured markers, and a new deformed model can be synthesized as follows. Given a new control point vector $\mathbf{c}$, for each bone b_j we select its influence control points $\mathbf{c}_j$ and search its k_p nearest neighbors among the example poses. The chosen poses each have a deformation model trained locally for bone b_j that can independently predict a bone rotation $\mathbf{d}^l_j(\mathbf{c}_j), l = 1, \ldots, k_p$. The final bone rotation is computed as a weighted average of the individual predictions:

$$\mathbf{d}_j(\mathbf{c}_j) = \sum_{l=1}^{k_p} w(\mathbf{c}_j, \mathbf{c}^l_j)\mathbf{d}^l_j(\mathbf{c}_j) / \sum_{l=1}^{k_p} w(\mathbf{c}_j, \mathbf{c}^l_j) \tag{3}$$

The weights $w(\mathbf{c}_j, \mathbf{c}^l_j)$ are calculated by the same heat kernel as in Equation (2). Note that bone rotations predicted by a local model $\mathbf{d}^l_j$ are defined with respect to the example pose $\widetilde{P}_l$. It is necessary to first transform them to the reference pose $\widetilde{P}_0$ before blending them. For the sake of notation simplicity we omit the coordinate transformation and directly write $\mathbf{d}^l_j(\mathbf{c}_j)$ in the above formula. Now from the predicted composite bone rotations, bone translations $\mathbf{t}_j(\mathbf{c}_j)$ can be computed via the Poisson solver as described in §5.1.3. Bone transformations $\mathbf{T}_j(\mathbf{c}_j)$ are then computed from $\mathbf{d}_j(\mathbf{c}_j)$ and $\mathbf{t}_j(\mathbf{c}_j)$. A vertex position is therefore calculable according to the skinning Equation (1).

The resulting mesh above is a low-resolution smooth mesh that approximates $\widetilde{P}_i$ and thus only reproduces large-scale deformations. To add details, we first subdivide the low-resolution mesh in the same way as in the subdivision procedure for detail extraction, and we also compute the normal $\mathbf{n}(v)$ for each vertex v. Then we use the trained displacement model $\mathbf{h}(\mathbf{c})$ to generate a high-resolution displacement map that can be added to the subdivided mesh along $\mathbf{n}(v)$. To put it into a mathematical form, a vertex position v in a synthesized model with details can be estimated as follows:

$$v(\mathbf{c}) = \sum_{j \in B(v)} w_j \mathbf{T}_j(\mathbf{c}_j)v^r + \mathbf{h}(\mathbf{c})\mathbf{n}(v) \tag{4}$$

where $B(v)$ stands for the influence bone set for vertex v.

Model	#Cpt	#Bone	DT(*min*)	Training(*min*) bone/disp.	Synthesis(*sec*) bone/disp.
Hand-I-K	17	1355	15	3/19	0.6/3.4
Hand-I-M	15	1355	15	3/19	0.6/3.4
Hand-II-K	19	1239	15	3/19	0.6/3.5
Hand-II-M	15	1239	15	3/19	0.6/3.5

Table 1: *Performance Statistics. '#Cpt': number of control points; '#Bone': number of bones; 'DT': time spent on deformation transfer; 'Training': training time for the bone deformation models and the displacement model respectively; 'Synthesis': synthesis time for bone transformations and vertex displacements. Hand-*-K: deformation modeling with the palm-side control points for keyframe animation; Hand-*-M: deformation modeling with the back-side motion captured markers for performance-driven animation.*

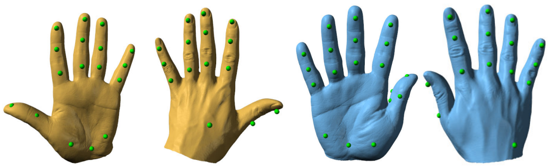


Figure 6: *Our testing models and control points.*

7. Experimental Results

We have implemented our deformation system in C++ on a 2.83*GHz* Intel Quad core machine with 8*GB* of RAM. The deformation results shown in the paper and the accompanying video are rendered by an OpenGL renderer with Phong shading.

Training Data Acquisition We use scanned hand models to demonstrate the capability of our deformation algorithms. Capturing hand models with fine wrinkles is difficult due to severe self-occlusions between fingers and the need of accurate registration for data captured from a not-entirely-static hand. We thus use the traditional art of body casting. Silicone rubber molds are first created from various hand poses. Then plaster models are casted from the silicone molds. We use the Konica-Minolta Range 7 laser scanner to scan the plaster hand models. The scanner can scan one region of a 3D object in about two seconds with high accuracy (±40*um*). The scanning software then processes and merges the point clouds and generates surface models. These models are at extremely high resolution of around 900K vertices. Experimentally we have found that downsampling these meshes to about 200K vertices does not lead to any visually noticeable difference [Tur92]. So we simply use the 200K meshes as our training examples.

We captured two hands from two male subjects, both graduate students in their twenties. Figure 6 shows our models at their reference poses. Hereafter we denote the left hand of the first subject as hand-I, and the right hand of the second subject as hand-II. For each hand, fourteen highly detailed mesh models are prepared as training data for deformation learning. We manually specify fewer than 20 mesh vertices

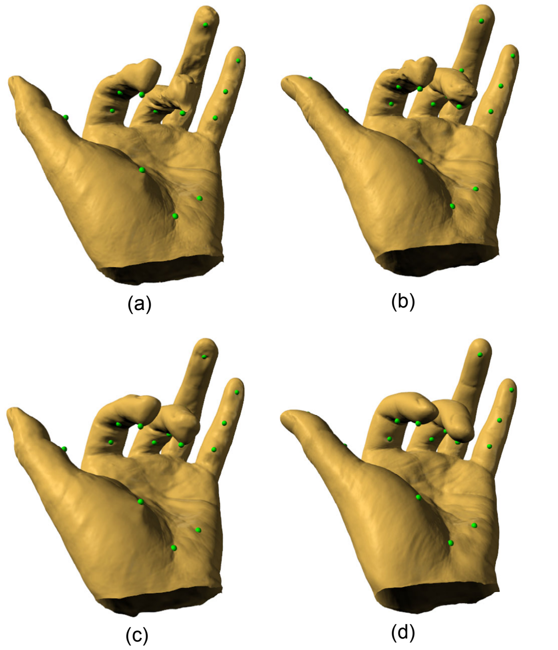


Figure 7: *Large-scale deformations generated from (a) globally trained models as in [FKY08]; (b) locally trained models in the geometry space; (c) locally trained models in the pose space; (d) locally trained models in both spaces.*

as control points as shown in Figure 6. There are two sets of control points defined for each hand: one set all on the palm, and the other all on the back. Our system works equally well with both sets of control points. The existence of two sets is basically a result of moving markers from the palm to the back of the hand during motion capture for testing the performance-driven capability of our system. Severe occlusions caused by finger movements for markers on the palm side make them unusable.

Performance Table 1 lists the performance statistics for each testing dataset with both control point sets. The timing of each stage of the deformation system is given. The model registration and training steps are done offline, but still within a reasonable timeframe of tens of minutes. Currently the large-scale deformation synthesis is interactive, and the detail synthesis is several-fold slower.

Validation and Comparison Figure 8 compares a mesh generated with our pose-driven displacement predictor to another mesh generated with a static displacement map extracted from the rest pose. The same bone transformation models are used for generating the deformed base mesh. This comparison shows that a displacement map extracted from one pose cannot reproduce the deformed mesh details in other poses. It also demonstrates that our displacement predictor well captures the variation of geometric details from the input examples.

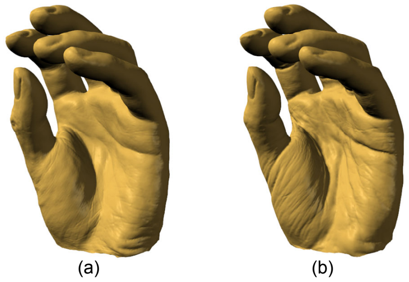


Figure 8: *Deformation results with a static displacement map (a) and our pose-dependent displacements (b).*

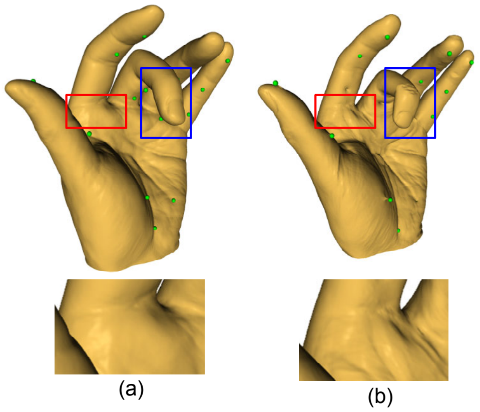


Figure 9: *Comparison with MeshIK. (a) Results from our method. (b) Results from MeshIK under the same control point configuration. The blue boxes point out one area that is problematic for MeshIK to synthesize correct large-scale deformations. The bottom row shows the close-up views of the area enclosed by the red boxes in the upper row. MeshIK produces false wrinkles in this case.*

Figure 7 demonstrates the effectiveness of our local modeling method as compared to the global regression approach of [FKY08]. Figure 7(a) exhibits large visual artifacts, while locally learned models in both the geometry space and the pose space effectively eliminate such artifacts as shown in Figure 7(d). How about only learning local models in either the geometry space or the pose space? Figure 7(b) and (c) show that local models in just one of the spaces are helpful in reducing the prediction errors, yet neither alone can completely eliminate all deformation artifacts.

Figure 9 compares our results with those of MeshIK [SZGP05] using the same control point input. Our results in Figure 9(a) predict natural novel poses with clean fine-scale details, while the overall pose and details synthesized by MeshIK contain visible artifacts. The close-up in Figure 9(b) further shows that details from multiple training poses incorrectly mix together, giving rise to unnecessarily cluttered wrinkles.

We also perform leave-one-out cross-validations for each example pose of each model, and report the average RMS

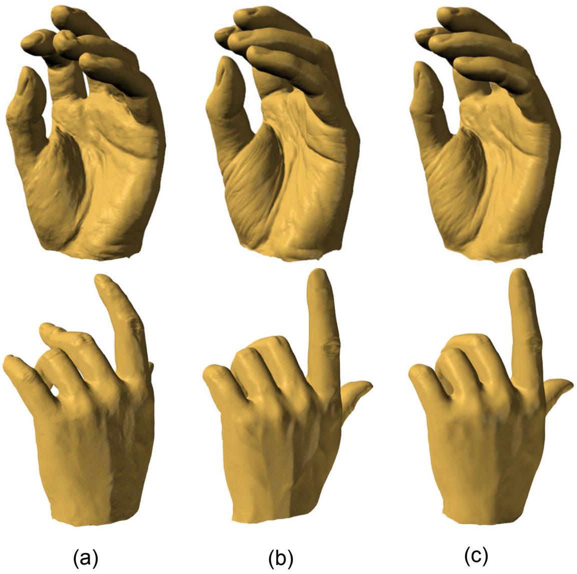


Figure 10: *Two leave-one-out cross-validations with (a) the method of [FKY08] and (c) our method. The ground truth, i.e., the models removed from the training examples, is shown in column (b).*

Model	#Example	#VertLow	#VertHigh	RMS Error	
				[FKY08]	Ours
Hand-I	14	49K	195K	0.0435	0.0315
Hand-II	14	49K	196K	0.0332	0.0240

Table 2: *Comparison of cross-validations between our method and [FKY08]. The RMS errors are the averaged results of the leave-one-out validation for each example pose. '#VertLow': vertex number of the low-res meshes $\widetilde{P}_i$; '#VertHigh': vertex number of the high-res meshes P_i.*

errors in Table 2. The largest dimension of the bounding box of each model is scaled to unit length. In Figure 10 we show visual comparison of two cross-validations using the hand-I model. Our method not only generates consistently lower prediction errors than the method of [FKY08], but also produces significantly better visual results, in terms of both the large-scale deformations and the fine-scale details. However, do note that the veins appear smoother in the bottom of Figure 10(c). Our deformation framework is fundamentally data-driven, and cannot synthesize features not present in the remaining training examples.

Keyframe Animation We developed a Graphical User Interface (GUI) for editing the position of control points. To provide fast visual feedbacks, we only compute and render the bone-based large-scale deformations during interactive editing. Once the user is satisfied with the large-scale deformations, displacement predictions are added to refine the results. Figure 12(a) and (c) show some representative frames of keyframe animation sequences of the two hands respectively. Despite the sparse examples used in the training phase, our deformation models produce smooth large-scale deformations of the whole hand as well as plausible deformations of detailed wrinkles for various gestures.

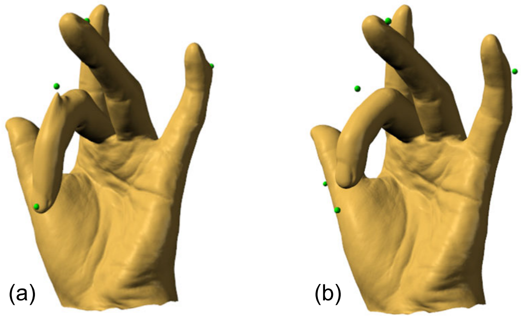


Figure 11: *Comparison of different weights for the control point constraint in the Poisson solver for noisy control points: (a) β=3.0, (b) β=0.3.*

Performance-driven Animation We also test our system with motion capture data. We place 18 motion capture markers on the back side of the hands, of which 15 are used to train deformation models. The remaining three markers plus one of the 15 markers used for model training are used for model registration. The 3D marker positions are then captured at $120Hz$ using a Vicon optical motion capture system, and then downsampled to $30Hz$ to drive our deformation models. Some selected frames are shown in Figure 12(b) and (d), although the results are best seen in the accompanying video. Our framework essentially provides a performance-driven animation system that can produce high-quality mesh animations with fine-scale details, using just a handful of captured marker trajectories. We showed our results to two artists, and some of their comments include:"The dynamic wrinkles look very realistic and consistent. Such quality is very hard to achieve manually"; "It would take us days to make one of such animation sequences using current commercial software packages".

There is one key difference in working with motion captured marker positions instead of manually edited control point trajectories. In an interactive editing setup, users generally want the pose to exactly follow their control points, and would otherwise feel frustrated. Yet the motion captured marker positions, if taken literally, can produce large deformation artifacts, such as that shown in Figure 11(a). There are many error sources degrading animations driven by captured markers. First, we cannot guarantee that the markers on the hand and the control points on the mesh are located at exactly the same spots. Second, the markers should be registered with the reference models to eliminate rigid transformations between them. Currently we use four markers placed at the back of the hand to estimate these transformations, but anatomically the back of the hand consists of many small bones and is not literally a rigid body. Third, there are noise and even missing markers caused by occlusions in the captured data. Lastly, to make our performance-driven animation system practical, we cannot assume that we can always motion capture the same hand from which we learned our deformation models. Indeed, our second subject was unable to participate in our motion capture sessions, and we had to capture the marker motions from a third subject to drive the hand-II model. Due to all the error factors above, for performance-driven animation we use a lower weight for the control point positional constraint term in the Poisson translation solver. Figure 11 illustrates this effect.

8. Conclusions and Discussion

Generating user-controllable mesh animation with rich details from sparse examples is a challenging open problem. Our main contribution is a robust framework that can produce fine-scale details as well as large-scale deformations by learning from extremely sparse training data. CCA-based regressions are used to model deformations of both layers, which not only makes the framework simple and clean, but also allows end-users to directly manipulate control points for interactive mesh animation. We demonstrate the effectiveness and robustness of our method using scanned hand models with edited or captured control point trajectories. Local fitting in both the geometry space and the pose space is the key to our success, which constrains models within the manifold of natural poses and effectively decouples independent object parts.

There are several limitations of the proposed method that deserve future investigation. First, we need to manually specify feature correspondences to initialize the deformation transfer algorithm for model registration. To ensure good registration for both large-scale and fine-scale features, we use 120 feature correspondences for the hand models. We have experimented with adapting an image-space optical flow algorithm to 3D surfaces to register models automatically. Our initial results indicate that registering features for highly deformable models with rich details is a tough problem. On a side note, 178 feature points are used for face registration in [MJC*08]. The registration method of [LAGP09] requires dense scans and would fail for our case as mentioned in their paper.

Our method is general in the sense that it does not require rigged example meshes. This enables direct utilization of scanned data. However, to better integrate with traditional skeletal animation tools and utilize legacy animations, it would be useful to drive deformable models using anatomically-based bone transformations. Another topic for future research concerns dynamic deformations. In this paper we primarily focus on pose-driven static deformations, which means there is a unique deformation associated with each pose. It would be interesting to investigate mechanisms to carry over deformation dynamics from previous instants of time.

Acknowledgements: This work was partially supported by National Science Foundation (IIS 09-14631). We would like to thank the anonymous reviewers for their constructive comments.

References

[BBA*07] Bickel B., Botsch M., Angst R., Matusik W., Otaduy M., Pfister H., Gross M.: Multi-scale capture of facial geometry and motion. *ACM Trans. Graph. 26*, 3 (2007), Article 33. 2

[BLB*08] Bickel B., Lang M., Botsch M., Otaduy M. A., Gross M.: Pose-space animation and transfer of facial details. In *SCA'08* (2008), pp. 57–66. 2

[BN01] Belkin M., Niyogi P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14* (2001), MIT Press, pp. 585–591. 5

[CD88] Cleveland W. S., Devlin S. J.: Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association 83*, 403 (1988), 596–610. 5

[dAST*08] de Aguiar E., Stoll C., Theobalt C., Ahmed N., Seidel H.-P., Thrun S.: Performance capture from sparse multi-view video. *ACM Trans. Graph. 27*, 3 (2008), Article 98. 3

[DSP06] Der K. G., Sumner R. W., Popović J.: Inverse kinematics for reduced deformable models. *ACM Trans. Graph. 25*, 3 (2006), 1174–1179. 2

[FKY08] Feng W.-W., Kim B.-U., Yu Y.: Real-time data driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. 27*, 3 (2008), Article 91. 2, 4, 5, 7, 8

[GMP*06] Golovinskiy A., Matusik W., Pfister H., Rusinkiewicz S., Funkhouser T.: A statistical model for synthesis of detailed facial geometry. In *SIGGRAPH'06* (2006), pp. 1025–1034. 3

[JT05] James D. L., Twigg C. D.: Skinning mesh animations. In *SIGGRAPH'05* (2005), pp. 399–407. 2, 4

[KCvO07] Kavan L., Collins S., Žára J., O'Sullivan C.: Skinning with dual quaternions. In *I3D'07* (2007), pp. 39–46. 5

[KJP02] Kry P. G., James D. L., Pai D. K.: Eigenskin: real time large deformation character skinning in hardware. In *SCA'02* (2002), pp. 153–159. 2

[KM04] Kurihara T., Miyata N.: Modeling deformable human hands from medical images. In *SCA'04* (2004), pp. 355–363. 2

[KV08] Kim T.-Y., Vendrovsky E.: Drivenshape: a data-driven approach for shape deformation. In *SCA'08* (2008), pp. 49–55. 2

[LAGP09] Li H., Adams B., Guibas L. J., Pauly M.: Robust single-view geometry and motion reconstruction. *ACM Trans. Graph. 28*, 5 (2009), Article 175. 3, 9

[LCF00] Lewis J. P., Cordner M., Fong N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH'00* (2000), pp. 165–172. 2

[LCXS10] Lau M., Chai J., Xu Y.-Q., Shum H.: Interactive manipulation of 3d facial expressions using facial priors. *ACM Trans. Graph. 29*, 1 (2010), Article 3. 4

[MG03] Mohr A., Gleicher M.: Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3 (2003), 562–568. 2

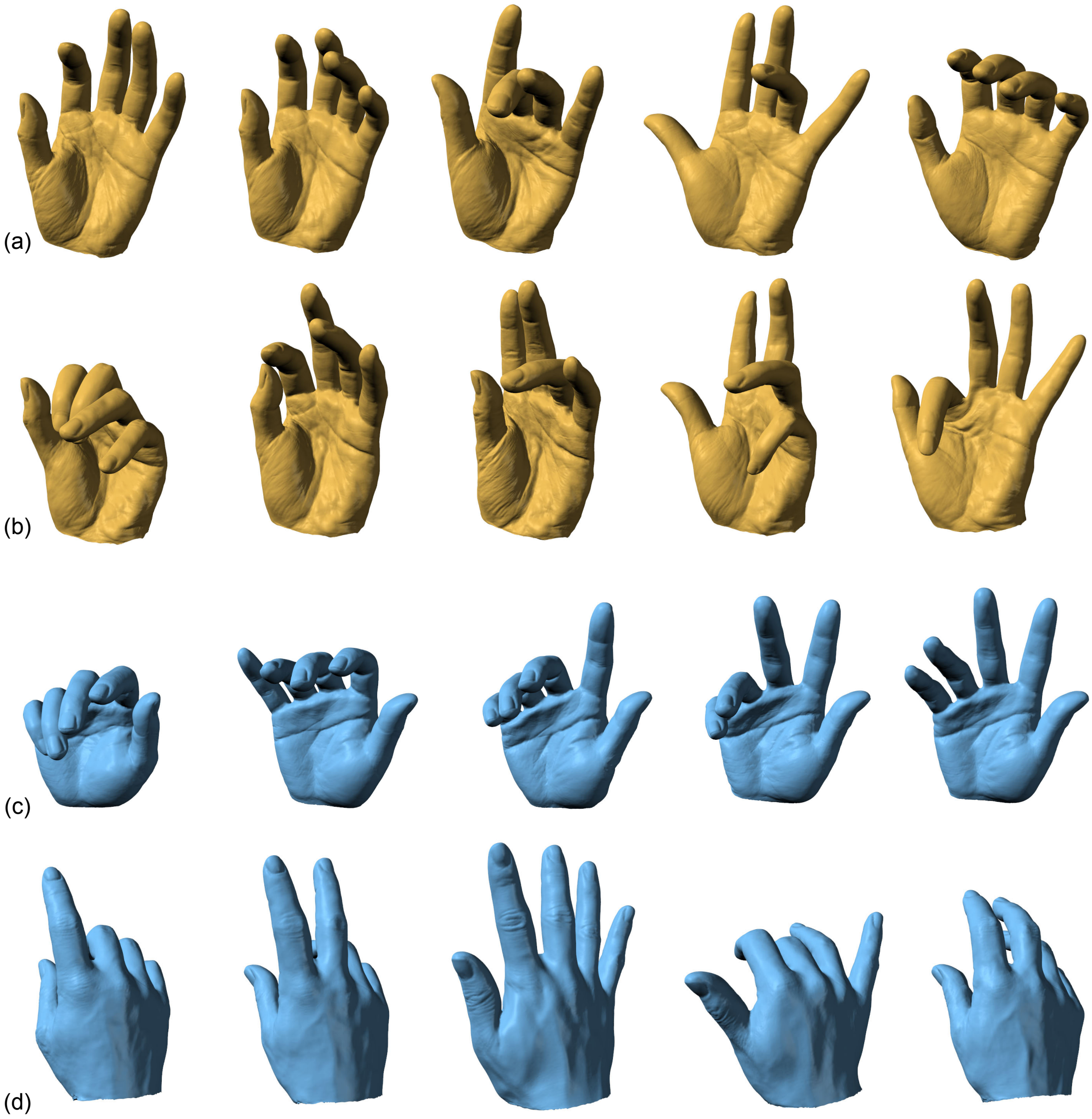


Figure 12: *Results of keyframe animation (a) and performance-driven animation (b) for the hand-I model; keyframe animation (c) and performance-driven animation (d) for the hand-II model.*

[MJC*08] MA W.-C., JONES A., CHIANG J.-Y., HAWKINS T., FREDERIKSEN S., PEERS P., VUKOVIC M., OUHYOUNG M., DEBEVEC P.: Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Trans. Graph. 27*, 5 (2008), Article 121. 2, 3, 9

[PH06] PARK S., HODGINS J.: Capturing and animating skin deformation in human motion. *ACM Trans. Graph. 25*, 3 (2006), 881–889. 2, 3

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (2004), 399–405. 3

[SR03] SAUL L. K., ROWEIS S. T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res. 4* (2003), 119–155. 5

[SRC01] SLOAN P.-P. J., ROSE C. F., COHEN M. F.: Shape by example. In *I3D'01* (2001), pp. 135–143. 2

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM Trans. Graphics 24*, 3 (2005), 488–495. 2, 7

[Tur92] TURK G.: Re-tiling polygonal surfaces. In *SIGGRAPH'92* (1992), pp. 55–64. 3, 6

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph. 27*, 3 (2008), Article 97. 3

[WPP07] WANG R. Y., PULLI K., POPOVIĆ J.: Real-time enveloping with rotational regression. *ACM Trans. Graph. 26*, 3 (2007), Article 73. 2

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Eurographics'07 26*, 3 (2007), 265–274. 2

[ZSCS04] ZHANG L., SNAVELY N., CURLESS B., SEITZ S.: Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph. 23*, 3 (2004), 546–556. 3