

Active Geometry for Game Characters

Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani

▶ **To cite this version:**

Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani. Active Geometry for Game Characters. MIG 2010 - 3rd International Conference on Motion in Games, Nov 2010, Zeist, Netherlands. pp.170-181, 10.1007/978-3-642-16958-8_17 . hal-00516412

HAL Id: hal-00516412
https://hal.science/hal-00516412

Submitted on 29 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Active Geometry for Game Characters

Damien Rohmer[1,2], Stefanie Hahmann[1], and Marie-Paule Cani[1,2]

[1] Grenoble Universités, Laboratoire Jean Kuntzmann
[2] INRIA.



Fig. 1. A sitting elephant with bulging belly and a jumping character with a wrinkling dress modeled using active geometric layers to reduce skin volume variations and cloth shrinkage.

Abstract. Animating the geometry of a real-time character is typically done using fast methods such as smooth skinning or coarse physically-based animation. These methods are not able to capture realistic behaviors such as flesh and muscles bulging with constant volume or fine wrinkling of animated garments. This paper advocates the use of active geometric models, applied on top of the current geometric layer, to mimic these behaviors without requiring any expensive computation. Our models fit into the standard animation pipe-line and can be tuned in an intuitive way thanks to their geometric nature.

Keywords: Geometry, Procedural model, Constant volume, Wrinkles.

1 Introduction

Animating characters with proper skin and clothing deformations is an open challenge in the field of computer games, where visual realism is desirable to

improve the users immersion while all computations are to be done in real time. In particular, skin deformations should not produce any unwanted loss of volume when the character articulates and cloth surface should not elongate or shrink during deformations. Although a physically-based simulation could be used to maintain these constraints, it would lead to stiff equations and would hardly be applicable in real-time. The deformation methods most often used in real-time applications are therefore: smooth skinning (SSD), despite of its well know artifacts; and very coarse, low-rigidity simulations when animating the dynamics of floating garments is required. These methods are not sufficient to get the flesh and muscle bulging behavior and the dynamic wrinkles one would expect.

This paper advocates the use of *active geometric models*, placed on top of the pre-existing layers, to add visual realism to animated characters at little extra cost. Active geometric models are procedural layers aimed at maintaining a given geometric constraint over time. They act by deforming and possibly refining the current geometry on the fly, just before rendering. We illustrate this concept by presenting two examples of such models, first introduced in [1,2]. The first one, based on volume control, is used to generate appropriate skin bulges and skin wrinkles when a character articulates. The second one, based on a measure of isometry with a rest shape, dynamically adds cloth-looking wrinkles automatically. Using these geometric models also eases user control: one just needs to specify deformation profiles for the constant-volume skin, and a single thickness parameter for the wrinkling cloth.

The remainder of this paper develops as follows: Section 2 is a brief state of the art on skin and cloth animation. Section 3 describes our method for enhancing SSD with volume constraints. Section 4 discusses the addition of plausible cloth wrinkles on top of a coarse simulated mesh used to animate floating garments. We conclude by discussing the results in Section 5.

2 Previous Work

2.1 Animating a character's skin

Smooth skinning deformation (SSD) is the standard method for skinning an articulated character in real-time: scalar weights relative to each bone of the articulated structure are associated with the mesh. Each mesh vertex is repositioned during animation using the weighted combination of the positions relatif to bone's local frame. This fast method unfortunately suffers from a number of artifacts, such as the loss of volume when a joint rotates or twists (see [3]).

Numerous attempts were made to improve SSD. A first family of approaches uses a pre-computation to best fit extended SSD parameters to some example pauses. The fitting can be performed on vector corrections [4], triangle deformations [5], matrices of influences [6], or on the position and weights of extra joints within the skeleton [7]. While such approaches enable to model complex local behavior of the mesh during animation, they require good skin models for designing a set of rest pauses to start with, and are limited to the range of motion

they span. Therefore unexpected motion in a video game may lead to unrealistic results.

A second family of approaches uses improved interpolation within the SSD framework at little extra cost. It includes rigidity constraints on the medial axis [8], non-linear matrix blending [9] or the use of the dual quaternions [10]. However no bulging effect (such as in fig. 1 left) due to the constant volume behavior of flesh and muscles can be modeled.

Closer to our goals, basic SSD was enhanced to get some dynamic behavior [11] and to generate wrinkling effects for skin or fitting garments [12], however without enforcing any constant volume or surface isometry constraint. Lastly, a constant volume extension of SSD was proposed [13]. It requires streamline integration of vertex positions and is thus time consuming and difficult to use in a standard animation pipeline. Contrary to this approach, we rely on standard SSD enhanced with constant volume skinning as independent post-processing corrections at each frame.

2.2 Animating character's clothing

Despite of a huge amount of research, realistic cloth simulation [14,15,16] still needs too much computing resources. Using coarse meshes or soft springs in mass-spring simulations reduces computation time but fails capturing typical cloth wrinkling behavior, since the surface tends to elongate or shrink rather than fold.

A number of geometric approaches were recently developed to model cloth material. The most recent ones [17,18] give impressive results but require training examples and can only model the deformations herein captured. The range of possible deformation is limited as well for pre-set procedural models such as the one based on cloth buckling around cylinders [19]. A subdivision process followed by vertex correction based on local minimization was used in [20] but requires a minimization step on highly refined meshes. Procedural wrinkle maps were investigated in [21,22] where wrinkle magnitude automatically adapts to triangle compression. However interpolating wrinkle directions on a texture map leads to unrealistic fading in and out of wrinkles during animation. Closer to our work, cloth wrinkles were defined as curves on the input cloth surface and used as deformers to generate animated wrinkles [23]. However curve shapes and influence radii were manually defined for a specific number of frames, requiring expert design skills and significant user time. Similarly to this approach, we encode wrinkles as surface curves, but use automatic algorithms to control their strength and their dynamic merging behavior over time.

3 Character skinning with volume control

This section describes the active geometric layer we add in top of standard SSD to get constant volume muscle and flesh deformation while providing some intuitive shape control. This work was first published in [1], which we refer to for more details.

3.1 Goals

Volume preservation for animated characters raises specific issues:

First, volume preservation should be local, as the human body does not inflate far away (as a balloon would do) when locally compressed. The loss of volume due to SSD typically takes place near articulations and should be restored there.

Second, the volume correction provided by our new active geometric layer should allow some local shape control, since muscles and fatty tissue can require quite different bulging profiles, from a smooth inflation to some wrinkling profile.

Our solution is a weighted volume correction step where the weights specify how much restoring a given local volume variation should affect the positions of the different mesh vertices. These weights can be pre-computed from the skinning weights as in [24] or be more precisely specified by distributing the amount of correction along the axes of bone-based local frames, thanks to 1-dimensional shape profiles.

3.2 Setting the volume to a fixed value

The volume enclosed in a closed triangular mesh $\overline{S}$ can be expressed as a trilinear function of its vertices $\overline{\mathbf{p}}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i = 1, \ldots, N$, as:

$$V = \sum_{\text{face}(l,m,n) \in S} \frac{z_l + z_m + z_n}{6} \begin{vmatrix} (x_m - x_l) & (y_m - y_l) \\ (x_n - x_l) & (y_n - y_l) \end{vmatrix}, \tag{1}$$

where $\text{face}(l, m, n) = \Delta(\overline{\mathbf{p}}_l, \overline{\mathbf{p}}_m, \overline{\mathbf{p}}_n)$ is a triangle of $\overline{S}$. Let S be the pointwise deformed surface mesh with vertex vector $\mathbf{p} = (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z) \in \mathbb{R}^{3N}$. The volume correction step consists of computing a correction vector $\mathbf{u} = (\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z) \in \mathbb{R}^{3N}$ applied to S such that the volume $V(\overline{\mathbf{p}}) = V(\mathbf{p} + \mathbf{u})$ of the corrected mesh equals the specified target value V_{target}. Note that any change of volume could be animated as well by prescribing a varying target volume over time.

We achieve volume correction by solving the constrained weighted least-squares problem:

$$\begin{cases} \min & \|\mathbf{u}\|_\gamma^2 \\ \text{subject to} & V(\mathbf{p} + \mathbf{u}) = V_{target}. \end{cases} \tag{2}$$

where $\|u\|_\gamma^2 := \sum_{k=1}^{N} u^2/\gamma_k$, γ_k. $\gamma = (\gamma_1, \ldots, \gamma_N)$ is a set of weights which specify the distribution of the correction over the mesh vertices.

To efficiently solve equation (2), the volume constraint is decomposed and processed separately on the 3 axes x, y and z, which gives a linear expression to each constraint [25], as follows:

First, u_x is computed as a solution of (2), but where u_y, u_z are set to fixed values. The constraint thus becomes linear. We derive an analytical expression of the solution using Lagrangien multipliers: $\mathbf{u}_x = \mu_0 \Delta V \frac{\nabla_{\mathbf{x}} V}{\|\nabla_{\mathbf{x}} V\|_{1/\gamma}^2}$, where $\Delta V := V(\mathbf{p}) - V_{\text{target}}$. The scalar value μ_0 determines the percent of volume correction in x-direction. Iterating this correction step in $y-$ and $z-$direction, with μ_1 and μ_2 percent respectively of the volume correction (where $\mu_0 + \mu_1 + \mu_2 = 1$)

insures that the final surface restores the target volume. The weighted closed form solution for the correction vector $\mathbf{u}_k$ at vertex k is finally given by

$$\mathbf{u}_k = (u_x^k, u_y^k, u_z^k) = \gamma_k \Delta V \left(\mu_0 \frac{\nabla_{\mathbf{x}} V(\mathbf{p}_k)}{\|\nabla_{\mathbf{x}} V\|_{1/\gamma}^2}, \mu_1 \frac{\nabla_{\mathbf{y}} V^\star(\mathbf{p}_k)}{\|\nabla_{\mathbf{y}} V^\star\|_{1/\gamma}^2}, \mu_2 \frac{\nabla_{\mathbf{z}} V^{\star\star}(\mathbf{p}_k)}{\|\nabla_{\mathbf{z}} V^{\star\star}\|_{1/\gamma}^2} \right), \quad (3)$$

where $\nabla_{\mathbf{x}} V(\mathbf{p}_k)$ designates the k^{th} entry of the gradient vector $\nabla_{\mathbf{x}} V$ and analogously for y and z. $\nabla_{\mathbf{x}} V$ is obtained by differentiating (1). V^* and V^{**} denote the volumes of the intermediate surfaces obtained after correction in $x-$ and $y-$direction respectively.

3.3 Getting organic-looking volume corrections

The volume correction method we just presented can be easily tuned to generate a variety of effects from muscle-like local bulges when a bone articulates to fatty-tissue looking bulges with folds and creases when the volume needs to be restored in a large compressed region such as the elephant belly.

While the mesh vertices weights γ are used to distribute the correction of a volume loss due to the action of a given bone, we also use the scalars (μ_0, μ_1, μ_2) introduced above to improve control over the direction of the volume correcting deformation.

This is done by successively applying the method above in local frames $(\mathbf{e}_0^b, \mathbf{e}_1^b, \mathbf{e}_2^b)$ associated with each bone b of the skeleton as illustrated in fig. 2 Left. This allows volumes variations due to the action of each bone to be processed one after the other, and the amount of correction $(\mu_0^b, \mu_1^b, \mu_2^b)$ (in percent) along the three axes to be controlled individually for each bone.

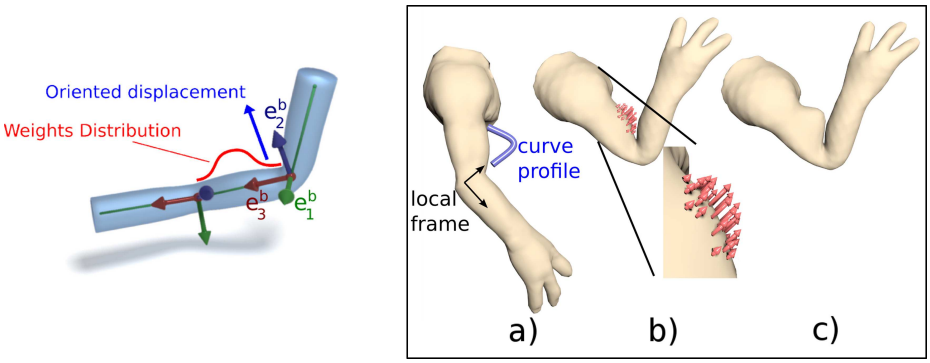


Fig. 2. Left: Local frames defined for an articulated cylinder deformed by smooth skinning using three segments. Right: Muscle effect using unidirectional inflation: a) A curve profile is designed in the local frame associated to the bending joint. b)Application of classical SSD. c)Volume correction vectors are computed using eq.(3).

In practice, the set of mesh weights in each local frame is controlled through user-specified 1D profile curves. They can be defined independently for each of the three axes. An intuitive control of the final deformation induced by volume correction is thus provided.

Selecting $\mu_2 \in [0.5, 1]$ to act only in the positive direction of a given axis of the local frame enables to mimic muscle effect as illustrated in fig. 2a-c.

More *c*omplex shapes can be modeled by using more complex profile curves or by combining them. For instance, fold effects can be achieved using oscillating curves. This can be obtained by setting $\gamma_k(\mathbf{p}_k^L) = \sin^2(\omega\pi x_k^L)\exp(-(\|\mathbf{p}_k^L\|/\sigma)^2)$, where ω modulates the frequency of the oscillations. In the cases of belly bulges of the elephant, in fig. 1 Left, we choose $\omega = 2$.

4 Active geometry for wrinkling cloth

This section explains how a low quality cloth animation can be quickly enhanced by plausible, animated cloth-like wrinkles. This work was first presented in [2], which we refer to for details.

In the following the appearance of the generated wrinkles is controlled through a single, intuitive parameter: the smallest radius of curvature $R_{\min}$ that the cloth can exhibit, which is linked to the thickness and stiffness of the fabric the user wants to mimic.

4.1 Computing shrinkage over an animated cloth mesh

Suppose we have some mesh animation for cloth, either generated using a coarse physically based simulation or any other deformation method such as SSD. Let $\overline{S}$ be the cloth mesh at rest, before motion starts. Ideally, the cloth deformation should preserve isometry with respect to this initial shape. Therefore, almost no stretch (compression or elongation) should occur between $\overline{S}$ and any deformed versions S of this mesh during the animation sequence. The idea is to compute the main shrinkage direction as a continuous vector field, from the information of triangle stretch from $\overline{S}$ to S. Then, wrinkles will be generated orthogonally to the local directions of compression. The framework of this method is illustrated in fig. 3.

To measure triangle stretch, we express the 2D affine transformation from a triangle in $\overline{S}$ to its counterpart in S by the 2×2 matrix T. Let $(\mathbf{u}_1, \mathbf{u}_2)$ and $(\overline{\mathbf{u}_1}, \overline{\mathbf{u}_2})$ be the 2D edge vectors of the triangle in the local frames of S and $\overline{S}$ respectively. Then the transformation matrix T is given by $\mathrm{T} = [\mathbf{u}_1, \mathbf{u}_2][\overline{\mathbf{u}_1}, \overline{\mathbf{u}_2}]^{-1}$. The 2×2 positive definite matrix $\mathrm{U} = \sqrt{\mathrm{T}^T\mathrm{T}}$ called *stretch tensor* measures the amount of compression and elongation conveyed by the transformation T. If λ, the smallest eigenvalue of U, is such that $\lambda < 1$, the triangle is compressed along the corresponding eigenvector $\mathbf{e}$ (fig. 3b).

As cloth should wrinkle orthogonally to local directions or compression, we use interpolation to define a continuous *wrinkle vector field* $\mathbf{v}$ over the mesh as

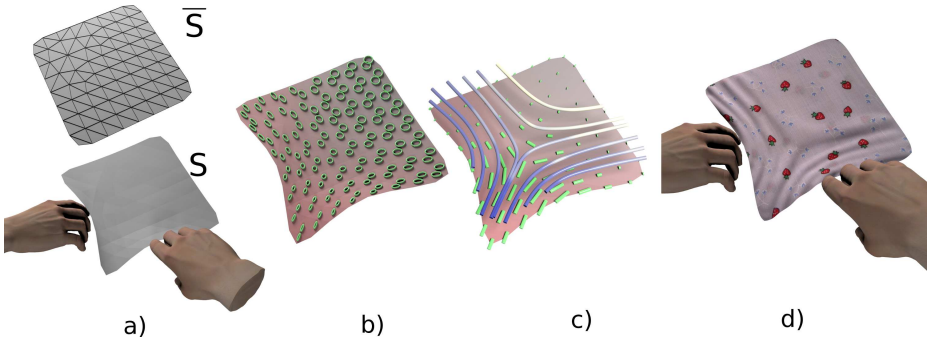


Fig. 3. Our active model for wrinkling cloth: a) Coarse input mesh and its initial, flat shape. b) Stretch tensor field. c) Wrinkle vector field $\mathbf{v}$ in green and the associated wrinkle curves. d) Final wrinkled mesh.

$\mathbf{v} = \max(1 - \lambda, 0)\ \mathbf{e}$, where $\|\mathbf{v}\|$ measures the amount of compression per unit of length.

This allows to define wrinkle curves as streamlines of $\mathbf{v}$: seed points are iteratively generated at vertices of maximal compression (while insuring that the distance between two of them remains larger than $2R_{\min}$) and streamlines are integrated in both direction from these points, until compression is smaller than a threshold.

4.2 Animating fold curves

Although this algorithm generates plausible wrinkle curves for a static frame (fig. 3d), it needs to be adapted to get temporal coherence during animation, i.e. prevent that small wrinkles appear and disappear or quickly slide over the cloth surface from frame to frame.

Firstly, curve seeds are tracked through time using a particle based approach: Seed positions from the previous frame are considered as potential seeds in the next frame, after being displaced toward the direction of the compression gradient. If the compression at the new location for the seed is still large enough, a wrinkle curve is generated there.

Secondly, the resulting wrinkle curve trajectories are smoothed over time to avoid discontinuities coming from the streamline integration process.

4.3 Wrinkle geometry

The animated wrinkle curves now have to be used as deformers to generate the final wrinkled cloth geometry. Our previous work [2] described a solution for generating a fine folded mesh requiring re-triangulation. Here, we rather discuss how this solution can be modified to generate a displacement texture, easier to use in the context of real-time animation and directly applicable on GPU.

Wrinkle parameters The wrinkle shape is controlled by two parameters: a curvature radius $R(u)$ along the curve wrinkle and the wrinkle offset $\beta(u)$ determining the portion of the circular arc used for forming the wrinkle. We compute $R(u)$ from the norm of the wrinkle vector field as: $R(u) = (1 - 2/\pi)/\|\mathbf{v}(u)\|$. Once the radius is set, the wrinkle offset is computed such that the shrinkage with respect to the rest shape is minimized. Fig. 4(left) summarizes the geometric relationship between $\beta(u)$ and $\|\mathbf{v}(u)\|$.

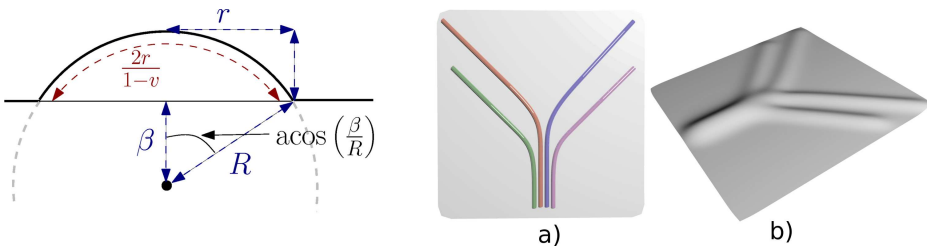


Fig. 4. Left: Relationship between width r, height h and the arc-length of a wrinkle generated at the offset distance β. Right: using wrinkle curves (a) as skeletons for an implicit deformer generates wrinkles which smoothly blend when they are close enough (b).

Wrinkle primitives The wrinkles we generate should smoothly merge or split during animation, depending on the distance between them. We choose to model them as implicit deformers in order to handle seamlessly this behavior.

The wrinkle curves are used as skeletons generating a field function f, to which the wrinkle surface is the iso-surface $f(\mathbf{p}) = 1$. Among the implicit surface models, we use convolution surfaces for their ability to model smooth primitives along polylines: f is defined using convolution of the Cauchy kernel along the skeleton. This specific kernel provides a closed-form solution to the convolution integral (see [26] for details).

The final field function of the implicit deformer is computed by summing the fields generated by the individual wrinkle curves (which will smoothly blend them where needed) and adding an extra field representing the current, coarse cloth mesh as well, for the wrinkles to smoothly blend into the cloth surface. To do so in an efficient way, the field generated by the mesh is approximated by the field generated by the tangent plane at the mesh point which is the closest to each query point.

Finally, mesh deformation is expressed as a displacement texture map which can be stored on the cloth patterns (see fig. 5). To compute such a map, sampled positions on the original 3D mesh are projected along their normal direction onto the isosurface of isovalue 1. The amount of displacement is then stored on the

texture map. Over-sampling is performed along the *wrinkle curve* location to ensure that the fine wrinkles are well captured by the texture.

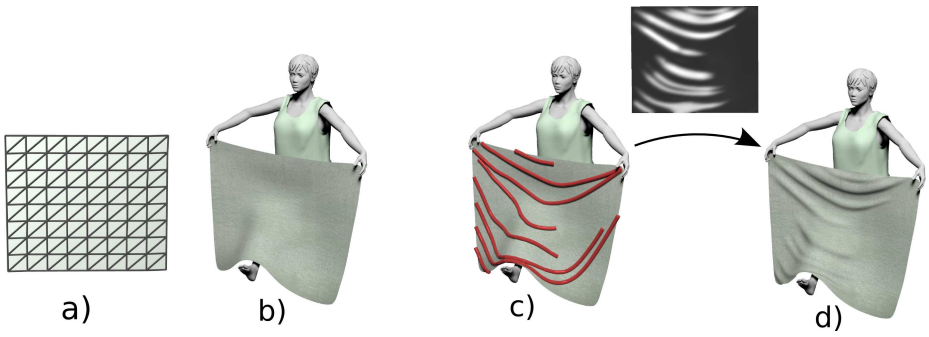


Fig. 5. Wrinkling a towel. a) Rest pose mesh. b) Deformed mesh using mass-spring system. c) Computed wrinkle curves. d) Final wrinkles generated using displacement texture map.

5 Results and Conclusions

Our geometric models for constant volume and wrinkling cloth have been tested on a variety of animated characters such as a sitting elephant and a human hand for volume control and bending, twisting and jumping of dressed characters for the cloth (fig. 1 and 6). Fast physical simulation using Blender software was used to generate the input coarse cloth meshes.

No effort was made to optimize the code in the current version of the systems: the average frame rate for constant volume skinning is about 0.2 seconds per frame, while it is about 2 seconds per frame for our cloth wrinkles (when a refined mesh is generated for the wrinkled surface). However, the current results could be optimized. In particular, since computations are independent on each mesh vertex, GPU optimization should be possible for both methods and would bring a significant improvement. In addition, the volume correction part could also be made more efficient when the exact volume constraint is not required. In such a case, approximated partial volume computations for each bone could be used instead of computing the volume exactly on the gobal mesh.

In conclusion, the active geometric layers we have been advocating throughout this paper are efficient solutions for adding plausibility to an animation through geometric constraints. Their expression as an extra module added on top of a pre-existing animation makes them easy to tune, allows the use of intuitive parameters (such as profile curves for bulging flesh or min radius for cloth wrinkles) and insures that they remain compatible with a standard animation

pipeline. We are currently experimenting with a third example of active geometric layer: some adapted convolution surface model used on top of a hair-guides simulation to animate volumetric, Manga-style hair with adequate wisps merging and splitting.

References

1. D. Rohmer, S. Hahmann, and M.-P. Cani. Exact volume preserving skinning with shape control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 83–92, August 2009.
2. D. Rohmer, T. Popa, M.-P. Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. *to appear in ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH ASIA*, December 2010.
3. J. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. *SIGGRAPH*, pages 165–172, 2000.
4. P. Kry, D. James, and D. Pai. Eigenskin: Real time large deformation character skinning in hardware. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 153–159, 2002.
5. R. Wang, K. Pulli, and J. Popovic. Real-time enveloping with rotational regression. *to appear in ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 26(3), 2007.
6. X. Wang and C. Phillips. Multi-weight enveloping: Least-squares approximation techniques for skin animation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 129–138, 2002.
7. A. Mohr and M. Gleicher. Building efficient, accurate character skins from examples. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 22(3), 2003.
8. J. Bloomenthal. Medial-based vertex deformation. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 147–151, 2002.
9. M. Alexa. Linear combination of transformations. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 21(3), 2002.
10. L. Kavan, S. Collins, J. Zara, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4), 2008.
11. C. Larboulette, M.-P. Cani, and B. Arnaldi. Dynamic Skinning: Adding real-time dynamic effects to an existing character animation. *Spring Conference on Computer Graphics (SCCG)*, 2005.
12. C. Larboulette and M.-P. Cani. Real-time dynamic wrinkles. *Computer Graphics International (CGI)*, 2004.
13. A. Angelidis and K. Singh. Kinodynamic skinning using volume-preserving deformations. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 129–140, 2007.
14. K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 21(3), 2002.
15. E. English and R. Bridson. Animating developable surfaces using nonconforming elements. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 27(3), 2008.

16. B. Thomaszewski, S. Pabst, and W. Strasser. Continuum-based strain limiting. *Computer Graphics Forum. Proocedings of Eurographics*, 28(2), 2009.
17. H. Wang, F. Hecht, R. Ramanoorthi, and J. O'Brien. Example-based wrinkle synthesis for clothing animation. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 29(4), 2010.
18. E. de Aguiar, L. Sigal, A. Treuille, and J. Hodgins. Stable spaces for real-time clothing. *ACM Transactions on Graphics (TOG), Proceedings of ACM SIGGRAPH*, 29(4), 2010.
19. P. Decaudin, D. Juilius, J. Wither, L. Boissieux, A. Sheffer, and M.-P. Cani. Virtual grarments: A fully geometric approach for clothing design. *Computer Graphics Forum. Proceedings of Eurographics*, 25(3), 2006.
20. M. Muller and N. Chentanez. Wrinkle meshes. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2010.
21. S. Hadap, E. Bangerter, P. Volino, and N. Magnenat-Thalmann. Animating wrinkles on clothes. *IEEE Proceedings on Visualization*, pages 175–182, 1999.
22. S. Kimmerle, M. Wacker, and C. Holzer. Multilayered wrinkle textures from strain. *VMV*, pages 225–232, 2004.
23. L. Cutler, R. Gershbein, X. Wang, C. Curtis, R. Maigret, L. Prasso, and P. Farson. An art-directed wrinkle system for CG character clothing. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2005.
24. D. Rohmer, S. Hahmann, and M.-P. Cani. Local volume preservation for skinned characters. *Computer Graphics Forum, Proceedings of Pacific Graphics*, 27(7), October 2008.
25. G. Elber. Linearizing the area and volume constraints. *Technical Report, TECHNION Israel*, 2000.
26. J. McCormack and Sherstyuk. Creating and rendering convolution surfaces. *Computer Graphics Forum*, 17, 2001.

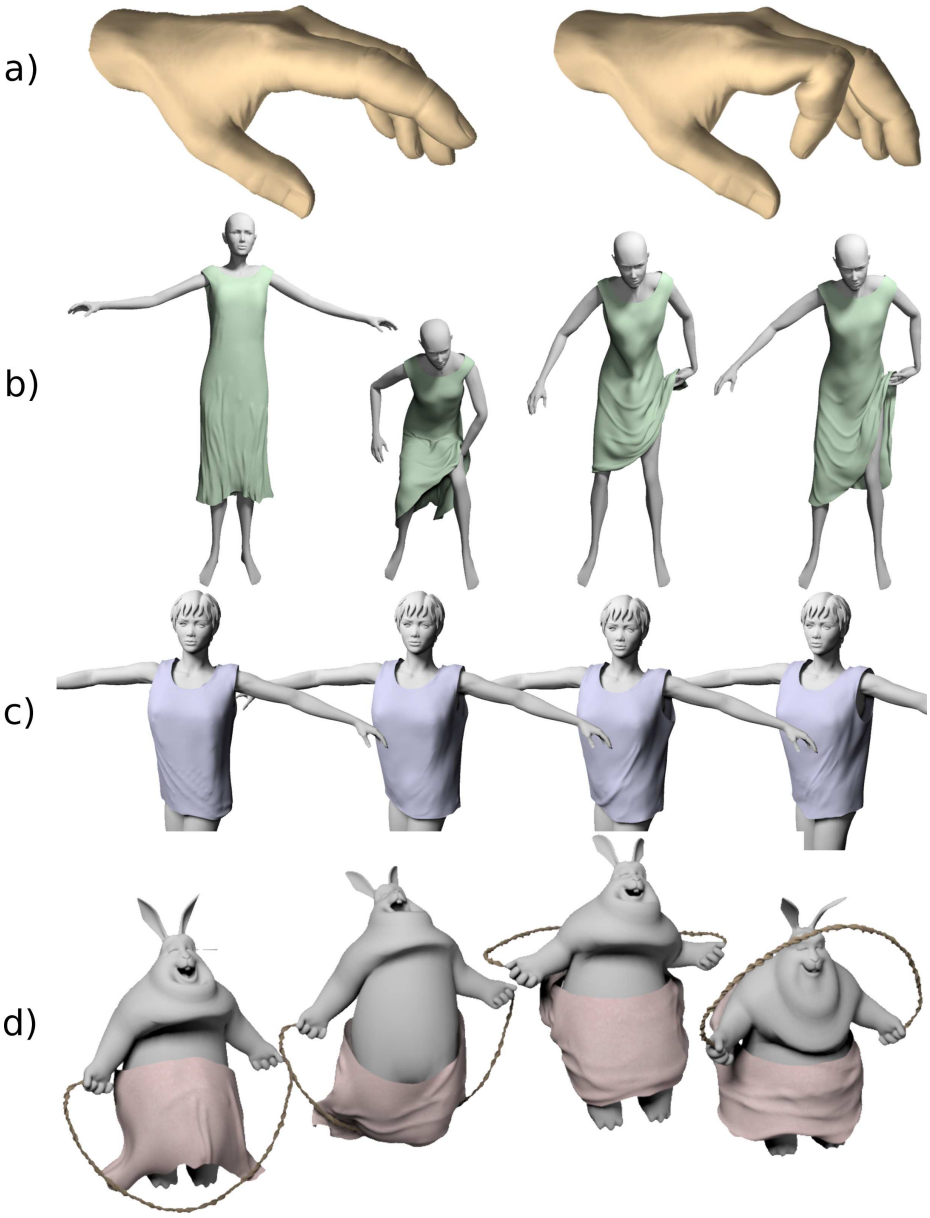


Fig. 6. a: Constant volume skinning of a human hand. Note the flesh bulges near the bent joint. b-d: Our wrinkles added on top of three coarse physically based animations of a human model and a bunny. Note the fine wrinkles added by our method where the coarse cloth model was shrinking.