Real-Time Dynamic Wrinkles

Caroline Larboulette
GRAVIR* & SIAMES-IRISA†, France
Caroline.Larboulette@imag.fr

Marie-Paule Cani
GRAVIR*, France
Marie-Paule.Cani@imag.fr

Abstract

This paper proposes a new method for designing dynamic wrinkles that appear and disappear according to the underlying deformation of tissues. The user positions and orients wrinkling tools on a mesh. During animation, geometric wrinkles are generated in real-time in the regions covered by the tools, mimicking resistance to compression of tissues. The wrinkling feature can be added to any existing animation. When the local resolution of the mesh is not sufficient, our tool refines it according to the wrinkle's finest feature. As our results show, the technique can be applied to a variety of situations such as facial expression wrinkles, joint wrinkles or garment wrinkles.

1. Introduction

Although well designed and animated in recent $3D$ animation movies [3], digital humans are not fully satisfying since they lack visually important details like wrinkles. Apart from some expression wrinkles, modeled with time and effort, their skin remains smooth whatever their body deformations. This is particularly annoying near joints, when the wrist bends for example. It's also a problem in the animation of virtual garments where physically-based simulation is too costly to be applied in real-time to every piece of cloth or to a fine mesh, especially in video games.

One can classify wrinkles into two different categories: *static* and *dynamic* wrinkles. The first ones are present on the skin regardless of movement and may only change over a lifetime (aging wrinkles case). Wrinkles of the other category, which we call *dynamic wrinkles*, rather depend on the current deformation of tissues. Skin or cloth being only compressible to a small extend, these wrinkles typically appear to absorb length changes, such as on the forehead when frowning, or on clothes when a joint bends (see fig 1).

Creating dynamic wrinkles is currently a tedious task for computer artists. Using softwares such as Maya [1] or 3ds Max [2], most of them carefully model the wrinkles for different poses of the skeleton. The current frame is then interpolated through the keyshapes. A deformer [1] may also be used. It consists in applying a deformation law to each selected vertex or group of vertices. Both techniques are tedious and time consuming. Moreover, if the artist has to modify a wrinkle, for instance move it a little bit along the mesh, almost everything needs to be redone.

(a) (b) (c)

Figure 1. Front (a) and back (b) of real jeans. (c): back of jeans simulated with our tool.

Our first goal is to enable the designer to achieve very easily at least comparable results. We allow a much quicker specification and editing of wrinkles by modeling them as an extra layer one can add onto a character model, without changing it or editing its animation. At each animation step, the wrinkling effects are computed from these specifications, and used to add an adequate deformation to the local geometry just before rendering. If necessary, wrinkles will yield a local refinement of the underlying mesh. Our second goal is to make our tool usable in computer games where real-time dynamic wrinkles are needed to enhance the animations. As wrinkling is a complex phenomenon that cannot be simulated in real-time if using physically-based models, we instead rely on geometric constraints mimicking physical properties, such as length preservation.

2. Related Work

In 1978, Blinn introduced bump mapping [5], which consisted in modifying surface normals before lighting calculations to give a visual impression of wrinkles, without

*GRAVIR is a joint lab of CNRS, Institut National Polytechnique de Grenoble, INRIA and Université Joseph Fourier.

†IRISA is a joint lab of CNRS, INRIA, INSA de Rennes and Université de Rennes 1

deforming the geometry. This idea has been extensively used since then, especially for facial wrinkles [13, 7]. Although being efficient and giving acceptable visual results in many cases, this technique suffers some drawbacks. As the geometry remains undeformed, the silhouette of objects is visually incorrect (noticeable in closed views) and no accurate collision detection in the wrinkled region is possible.

The first drawback of bump mapping can be solved using displacement mapping for rendering. For example, Volino [16] animates wrinkles on deformable models by modulating the amplitude of a given wrinkle pattern on a per triangle basis, with mesh refinement where necessary. This work was extended for cloth wrinkles in [10] with good visual results. However, as stressed by Kono [11], the tedious wrinkle pattern drawing and parameters tuning is left to the user. Bando [4] uses a more intuitive interface to obtain the displacement map. The user has to specify wrinkles one by one on the $2D$ projection of the $3D$ mesh by drawing a Bezier curve as the wrinkle furrow. However, a quite costly precomputation involving energy minimization is required to compute a specific mesh. The displacement or bump map may also be obtained using complex physically-based simulations [19, 18, 17, 7]. Whatever the technique used, the mesh geometry remains unchanged and thus prevents post-treatments such as collision detection.

Other existing techniques directly deform the geometry. For facial animation, Viaud [15] uses a mesh where the locations of potentially existing wrinkles are aligned with isolines of a spline surface. Combaz [8] generates complex folded geometry by simulating the static deformations of a finite element mesh by an internal growth process. An intuitive interface for painting the main wrinkle directions and their frequency is provided. Results are convincing, but the process is not real-time and does not directly apply to our problem, since we aim at simulating a surface which wrinkles to resist to compression rather than wrinkles created by an expansion process.

Since dynamic wrinkles are due to the length conservation constraints inherent to physical tissues, our approach is to automatically generate them from these geometric constraints rather than using simulation or asking the user to carefully design them. Sauvage [14] proposed a model for multi-resolution curves that preserve their length during interactive manipulation by wrinkling at a predefined scale. This method is unfortunately far from real-time and does not directly apply on a mesh. Our multi-resolution wrinkling curve presented in section 3 preserves the length of its control polygon and directly controls the displacements of the mesh vertices (section 4).

3. Multi-resolution wrinkling curve

Our tool relies on a planar curve which dynamically wrinkles when its extremities get closer to each other. This curve defines a possible profile for surface wrinkles and is used to apply deformations over a $3D$ mesh.

Control Curve The wrinkle is animated thanks to a $2D$ discrete control curve of constant length. This curve is defined by two endpoints, the origin and the target, and by a rest length value l. At rest, the curve is a line segment containing n control points (fig. 2). When the endpoints come closer to each other, the positions of the control points along the curve are recomputed to keep the length constant.

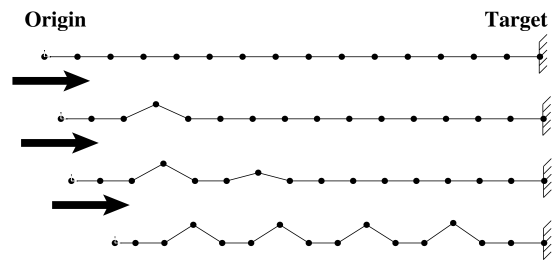

Figure 2. A control curve and its deformation when the origin comes closer to the target.

Deformation algorithm Before each rendering step,

1. the new length of the segment $\overrightarrow{\text{Origin Target}}$ is computed;
2. the control points are recomputed so that they remain equally spaced when projected on the x-axis;
3. some of the control points are moved in the y direction in order to preserve the original length of the curve.

Length conservation The idea is to re-inject the loss of length along the x-axis in the y direction. Let d be the shortening of the segment and x the distance on the x-axis between each point. As the points are equally spaced on the x-axis, we can easily compute the vertical displacement h of a control point needed to absorb the loss of length: $h = \sqrt{dx + \frac{d^2}{4}}$.

Wrinkling strategies In practice, the user can choose between different dynamic wrinkling strategies: bumps may propagate from the origin of the curve, from both endpoints, or they may appear simultaneously everywhere along the curve. In the two first cases, the maximal height h_{max} of a bump is specified. When h_{max} is reached, the wrinkle propagates. In the last case, the shortening d is divided by the number of bumps. All parameters including the space between bumps and the size of bumps are tunable.

Levels of details As we can see on fig. 3(a), real wrinkles often result in the combination of waves at different scales. To model this behavior, a portion of the length d to be re-injected in the curve is kept to create little wrinkles at a smaller resolution onto the top of the big ones. The control curve is subdivided to achieve the desired level of details, by iteratively multiplying the number of control points by two.

(a) (b)

Figure 3. Wrinkles at different levels of details: a real hand (a), with our tool (b).

Note that coarse to fine wrinkles can be defined independently from the resolution of the mesh, an automatic local refinement is processed when and where needed so that the wrinkling effect is correctly rendered (see section 4).

4. Creation of surface wrinkles

This section explains the creation of the wrinkling tool associated with the curve and its application onto a mesh.

Set up of the wrinkling tool The wrinkling curve serves as a tool to control the local deformations of an underlying mesh. The designer initializes a wrinkling region by drawing a line segment onto the mesh, perpendicularly to the desired orientation of the wrinkles. In order to deform during animation, the wrinkling curve is automatically anchored to the underlying mesh by attaching each endpoint to the nearest mesh vertex. The user then chooses an adequate wrinkling behavior by tuning the parameters to specify the way the curve deforms in the plane defined by the line segment and the mean direction between the two normals of the mesh at the anchoring points.

Region of influence A region of influence of a given width is associated to each wrinkling curve. It's a rectangular patch, centered on the initial curve segment (see fig. 4 left). The user chooses an attenuation profile which dictates the way the bumps decrease and then vanish when it goes away from the wrinkling curve. Attenuation values are always set between 1 at the center to 0 at the border of the patch. In our current implementation, two attenuation profiles, a linear one ($y = 1 - abs(x)$) and a bell shape one ($y = 1 + \frac{(-4x^6+17x^4-22x^2)}{9}$), are provided.

Mesh deformation The wrinkling tool directly controls a local deformation of the mesh that is updated before rendering, after each standard animation step. The vertices influenced by the wrinkling tool are all those that project onto the tool patch along its normal vector before deformation (see fig. 4). Each vertex displacement is computed from the (u, v) coordinates of its projection onto the patch during animation: the height given by the wrinkling curve at u is multiplied by the value of the attenuation function at v. This displacement is applied in the direction of the normal of the mesh vertex on the undeformed surface.

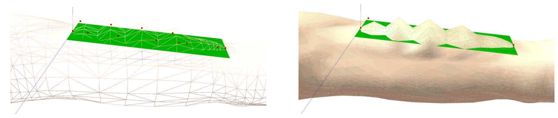

Figure 4. Wrinkles created by the contraction of the wrinkling curve and its influence patch.

Mesh subdivision To get good results, this method needs to be used on an adequately fine mesh, so that the dynamically wrinkled shapes are correctly captured. Asking the user to refine the mesh prior to animation would be both tedious and costly, since a fine mesh would be used for animation and rendering even when the wrinkles are not active. We thus add a local mesh subdivision step on the fly when the control curve shortens and where influenced triangles are too large according to the distance between two control points. We locally apply a modified butterfly scheme [9, 20], i.e. an interpolation scheme that does not smooth the mesh too much. Note that this is a relatively costly step but we keep interactive rates since the number of subdivision steps is usually small (1 or 2).

5. Applications and results

All the animations[1] shown in this section have been set up using keyframes to animate the skeleton and standard skinning [1, 12, 6] for the underlying mesh. They were captured in real-time on an AMD Athlon XP 1700+ under Linux, with a Radeon 9700 Pro graphics board.

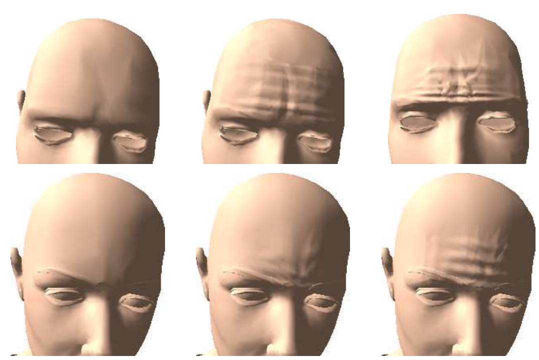

Figure 5. Wrinkles onto two different head models. Left: undeformed meshes. Middle and right: dynamic wrinkle simulation.

Forehead wrinkles Figure 5 depicts the application of dynamic wrinkles onto two different head models: a male model on the top line, and a female model on the bottom line. Since no existing facial deformation on which to attach the wrinkles was available to us, we mimicked the dynamic wrinkling effect by manually changing the rest length of the wrinkling curve. The simulation runs from 3.6 fps (2 subdiv.) to 13 fps (1 subdiv.) for a 21267 polygons mesh.

[1] http://www-imagis.imag.fr/Publications/2004/LC04

Joint wrinkles Figure 6 illustrates the use of our dynamic wrinkling tool to model wrinkles that appear near the wrist when it bends. Our tool automatically adds adequate details to skin deformation, which makes the overall motion more believable. The animation with wrinkles (fig. 6 middle and right) runs from 23 fps (2 subdiv.) to 188 fps (no subdiv.) for a mesh containing 2480 polygons.

Figure 6. Left: standard skinning animation. Middle and right: with our wrinkling tool.

Wrinkles on clothes More complex wrinkle shapes can be obtained by the combination of several wrinkles that run on top of each other. We can observe such wrinkles on trousers, in the knee region, as shown in fig. 1(a). The resulting animation (fig. 7), runs at 17 fps with 4 wrinkling tools and 1 subdivision level for a mesh of 6362 polygons.

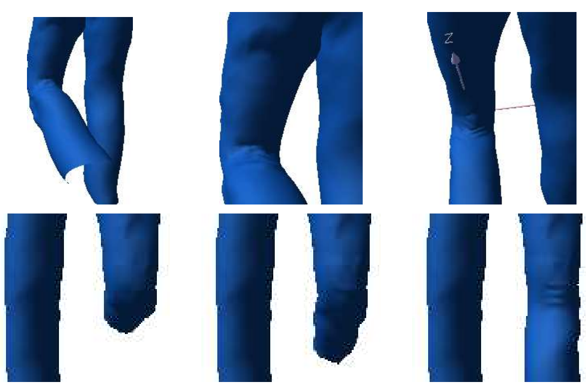

Figure 7. Back (top) and front (bottom) of virtual trousers with 2 overlapping wrinkles.

6. Conclusion and future work

We have presented a new and easy way to model and animate dynamic wrinkles on an existing model. Our procedural technique is based on geometric constraints (length preservation) to achieve visually realistic results without the heavy cost of a physically-based simulation. Wrinkles are added on top of a mesh and deform in real-time in response to its underlying motion. They do not require any manual modification of the underlying mesh or of the animation sequence, which saves time and effort to the computer artist.

We are currently studying an extension of this technique to create curved wrinkles caused by an interaction, such as when one touches the back of his own hand and makes the skin slide in a given direction. We are also planning to combine our wrinkle model with a simple dynamic muscles and fatty tissues model to give more life to the animation of the underlying mesh. A more long-term focus will be to detect the collisions due to the large wrinkles we generate and add an adequate resulting deformation to the contact region.

Acknowledgements

We'd like to thank S. Kimmerle and the WSI GRIS project of Tübingen University for providing the trousers model, C. Depraz for editing the models and ATI for providing the graphics board.

References

[1] Alias-Wavefront, Maya. http://www.aliaswavefront.com.

[2] Discreet, 3ds Max. http://www.discreet.com/3dsmax/.

[3] Mike Arias, The Animatrix. http://www.theanimatrix.com.

[4] Y. Bando, T. Kuratate, and T. Nishita. A simple method for modeling wrinkles on human skin. In *Pacific Graphics '02*.

[5] J. F. Blinn. Simulation of wrinkled surfaces. *Proceedings of SIGGRAPH'78*, pages 286–292, 1978.

[6] J. Bloomenthal. Medial-based vertex deformation. In *Proceedings of the ACM SIGGRAPH symposium on Computer animation*, pages 147–151. ACM Press, 2002.

[7] L. Boissieux, G. Kiss, N. Magnenat-Thalmann, and P. Kalra. Simulation of skin aging and wrinkles with cosmetics insight. In *Computer Animation and Simulation '00*, p. 15–27.

[8] J. Combaz and F. Neyret. Painting folds using expansion textures. In *Pacific Graphics*, Oct. 2002.

[9] N. Dyn, D. Levine, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, Apr. 1990.

[10] S. Hadap, E. Bangerter, P. Volino, and N. Magnenat-Thalmann. Animating wrinkles on clothes. In *IEEE Visualization '99*, pages 175–182, Oct. 1999.

[11] H. Kono and E. Genda. Wrinkle generation model for 3d facial expression. *Sketches and Applications, SIGGRAPH'03*.

[12] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH'00, ACM Computer Graphics*, pages 165–172, July 2000.

[13] S. Pasquariello and C. Pelachaud. Greta: A simple facial animation engine. In *Proc. of the 6th Online World Conference on Soft Computing in Industrial Applications*, Sept. 2001.

[14] B. Sauvage, S. Hahmann, and G.-P. Bonneau. Length preserving multiresolution editing of curves. *Computing*, to appear 2004.

[15] M.-L. Viaud and H. Yahia. Facial animation with wrinkles. In *EG Workshop on Animation and Simulation*, Sept. 1992.

[16] P. Volino and N. Magnenat-Thalmann. Fast geometrical wrinkles on animated surfaces. In *Seventh International Conference in Central Europe on Computer Graphics and Visualization (WSCG)*, Feb. 1999.

[17] Y. Wu, P. Kalra, L. Moccozet, and N. Magnenat-Thalmann. Simulating wrinkles and skin aging. *The Visual Computer*, 15(4):183–198, 1999.

[18] Y. Wu, P. Kalra, and N. M. Thalmann. Simulation of static and dynamic wrinkles of skin. In *Proceedings of Computer Animation '96*, pages 90–97, June 1996.

[19] Y. Wu, N. M. Thalmann, and D. Thalmann. A plastic-visco-elastic model for wrinkles in facial animation and skin aging. In *Pacific Graphics*, pages 201–213, 1994.

[20] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proc. of SIGGRAPH '96*, pages 189–192. ACM Press New York, 1996.