Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2010)
M. Otaduy and Z. Popovic (Editors)

Wrinkle Meshes

Matthias Müller & Nuttapong Chentanez

NVIDIA

Abstract

We present a simple and fast method to add wrinkles to dynamic meshes such as simulated cloth or the skin of an animated character. To get the desired surface details, we attach a higher resolution wrinkle mesh to the coarse base mesh allowing the wrinkle vertices to deviate from their attachment positions within a limited range. The shape of the wrinkle mesh is determined by a static solver which runs in parallel to the motion of the base mesh. Our method can be used to automatically enhance a purely animated skin mesh with wrinkles which would be a tedious task to do by hand. The fact that the tessellation of the wrinkle mesh can be chosen independently of the structure of the base mesh can be used to control the look of the wrinkles. The locations of wrinkle formation can be defined by painting the maximum distance the wrinkle mesh is allowed to deviate from the base mesh.
The second important application of wrinkle meshes is to add detail to simulated meshes such as cloth. Our method allows one to reduce the resolution of the simulation mesh without losing interesting surface detail. This speeds up the simulation, collision detection and handling and it reduces stretchiness. We show the efficiency and visual quality of the approach in a real-time setting.

1. Introduction

In real-time environments such as computer games, simulation time is limited to just a few milliseconds per time step. This prevents the use of high resolution meshes as in off-line simulations [BMF03, MTV05]. The coarse meshes that are typically used in games today do not show small scale detail making the cloth look stiff and unrealistic. Fortunately, wrinkle formation and the global dynamic motion of cloth can be split into two separate processes without introducing disturbing visual artifacts. In addition, wrinkling can be modeled as a quasi static phenomenon as wrinkles do not oscillate in typical scenarios.

Many researchers have recognized these facts and used them to derive methods for adding wrinkles to a low resolution simulation. With only a few exceptions, this is accomplished by adding high resolution displacements in the normal direction of the base mesh. We remove this restriction and allow the wrinkle mesh to deform in the tangential direction as well. We show that these additional degrees of freedom improve the look of wrinkles substantially. In short, our main contributions are

- To create wrinkles by attaching a separate quasi-statically deforming wrinkle mesh to a base mesh using Bézier interpolation.
- To use both the normal and tangential degrees of freedom to create wrinkles.
- To get realistic motion of the wrinkle mesh by applying a two phase force profile to the edges of the base mesh.
- Methods for collision handling.

Our method is easy to implement and almost fully automatic but still allows the user to control various aspects of it if desired.

2. Related Work

There is a large body of work on cloth simulation in general (see the survey paper [NMK*06] for instance). The majority of papers about wrinkle simulation propose to add high resolution displacement maps along the normals of the underlying base mesh. Hadap et al. [HBVT99] compute a bump map on a coarse mesh that approximately preserves area by blending several user-defined bump patterns. This idea was later extended by Kimmerle et al. [KWH04]. The technique can be implemented in shaders and is thus, fast enough to run in real-time. The same is true for the approach proposed by [RMB08]. They pre-compute normal maps that are blended in and out depending on the deformation of the base mesh. Loviscach [Lov06] blends procedurally generated sinusoidal wrinkle patterns based on the local deformations

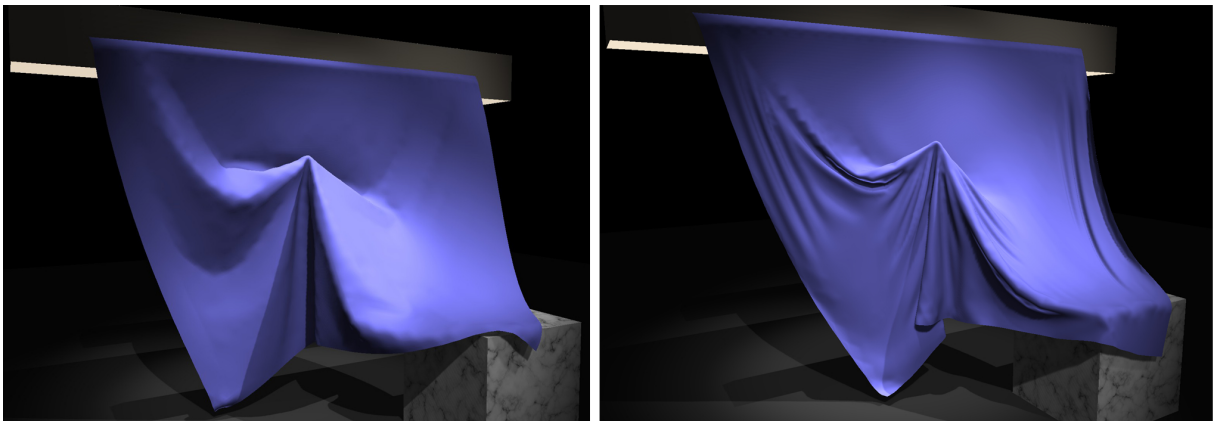

Figure 1: *Left: The simulated base mesh. Right: Wrinkle mesh attached to the base mesh. The reduced compression resistance of the base mesh makes the cloth look as if it was simulated in full resolution.*

of vertices in the base mesh. Larboulette and Cani [LC04] as well as Wang et al. [WWY06] employ the idea of edge length preservation to add wrinkles around a user defined line. Letting the user place the lines at arbitrary locations gives a high level of control of the locations and the orientations of wrinkles, but can also be tedious. Tsiknis [Tsi06] evaluates the displacements along normals by solving a per-triangle energy minimization problem. Recently, Eibner et al. [EFP09] proposed an approach to create folds for leather seat design. They procedurally generate 2d lines perpendicular to the global folds of the seat with certain pre-defined distributions.

Instead of creating the high resolution wrinkle data procedurally, another approach is to capture and store this information from either high resolution simulations [WHRO10] or motion capturing [BRW*07]. In those cases, the wrinkle displacements have to be stored in a data base. A problem of this approach in connection with computer games is the amount of memory needed to store the data. Also, the number of deformation modes of the base mesh has to be small and known in advance to prevent the data base from growing exponentially. This assumption holds for a face or tight fitting clothing on a character with a fixed skeleton structure. It does not hold for loose clothing or arbitrarily moving cloth though. Our approach can handle these cases well because the wrinkle data is computed on the fly.

In contrast to the papers above, Kang and Cho [KC02] use a second higher resolution mesh for wrinkle formation that has all degrees of freedom as in our approach. However, both meshes are dynamic and fully simulated with the exception that external forces are only applied to the coarse mesh. To keep the meshes synchronized, velocity changes are transferred from the coarse mesh to the fine mesh and back. In contrast to our method, there are no explicit links between fine vertices and the base mesh triangles to control wrinkle

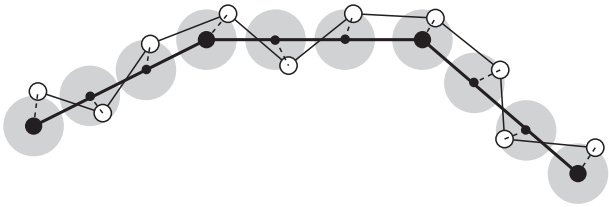

Figure 2: *2d sketch of the main idea: The large black dots and the thick black lines are the vertices and edges of the simulated or animated base mesh. The white dots and thin black lines represent the wrinkle mesh. Each white vertex is attached to the base mesh and allowed to move within the grey region to keep the fine edge lengths constant.*

formation and simplify collision handling. The authors also reduce the stiffness of the base mesh to increase wrinkle formation but in contrast to our method, they do not distinguish the two phases depicted in Fig. 6.

Our solver is basically a static version of the Position Based Dynamics [MHR06, Jak01] and related to the technique used in *Maya*'s *Nucleus*. As in [Mül08], our method helps to reduce stretchiness by employing coarse and fine meshes. In contrast to these approaches however, our method is fully decoupled from the main solver. It can therefore be added in various scenarios in a plug-and-play fashion without any modifications of the core engine. We have a GPU implementation that keeps the wrinkle mesh completely on the graphics card for simulation and rendering. The interface is simple. All the application has to do is to provide the positions and normals of the base mesh for creation and then at each time step.

3. Wrinkle Mesh Creation

Fig. 2 shows the main idea behind our algorithm. We start with a base mesh which can be a simulated mesh such as a piece of cloth or an animated mesh such as the skin of a rigged character. Our goal is to enhance the surface detail by adding wrinkle patterns. To accomplish this we use a second, higher resolution *wrinkle mesh*. This mesh can be tesselated independently of the tessellation of the base mesh. It can even have a different genus closing holes for instance. We compute references from the wrinkle vertices to the base mesh which are composed of the index of a base triangle and a pair of barycentric coordinates that define a point within the base triangle. Not all wrinkle vertices need to be attached to the base mesh although non-attached vertices should be sporadic in order to get a correct wrinkling behavior. In our examples, we work with attached vertices only. Experimenting with non-attached ones is part of our future work.

There are many ways to create a wrinkle mesh. A simple one is to start with the base mesh and keep applying edges splits until the longest edge falls below a given threshold (see for instance [MTG04]). The references to the base triangles and the corresponding barycentric coordinates can easily be determined during that process. When the base mesh is regular such as the mesh of a curtain or flag, it is desirable to keep the wrinkle mesh regular as well in order to minimize visual artifacts due to the tesselation. This can be achieved by subdividing each triangle regularly into a fixed number of similar sub-triangles. Again, determining the references is straightforward.

In order to have maximum control over the wrinkle patterns, the mesh can also be created by an artist alongside the base mesh. The artist controls wrinkle formation by increasing the resolution of the mesh in areas of interest. In this general case, the references have to be found by determining the closest point of each wrinkle vertex on the base mesh.

3.1. Constraints

We use three constraint types (see Fig. 3).

- *Attachment constraint*: Each wrinkle vertex is restricted to stay within a given radius r_i to its attachment point on the base triangle. We define this radius per vertex by painting the wrinkle mesh. The attachment constraints are important to prevent the wrinkle mesh from forming low frequency buckling patterns and can be used to control the frequency of the wrinkles.
- *Distance constraint*: The wrinkle vertices are restricted to keep the edge lengths equal to their rest lengths. We determine the rest lengths by measuring the edge lengths in the original configuration of the wrinkle mesh – the bind pose in case of animated characters – and multiplying that value with a *pre-stretch factor*, s. For $s < 1$ the wrinkle mesh is pre-stretched and wrinkles only appear in highly compressed regions while for $s > 1$, wrinkles form even in the rest state. In the example shown in Fig. 8 we painted s on a per edge basis explicitly while we used $s = 0.9$ globally in the clothing scenes.
- *Bending constraint*: To control the bending stiffness of the wrinkle mesh, we add additional edge constraints connecting opposite vertices in pairs of adjacent triangles. Working with dihedral angles would be another possibility. We found the simple and fast distance constraint approach to be sufficient in our use cases though.

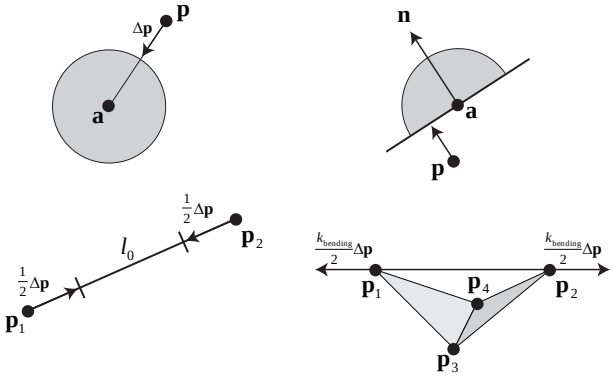

Figure 3: *Constraints on the wrinkle mesh. Top left: Attachment constraint. Top right: One sided attachment constraint to avoid collisions. Bottom left: Distance constraint. Bottom right: Simplified bending constraint with stiffness $k_{bending}$.*

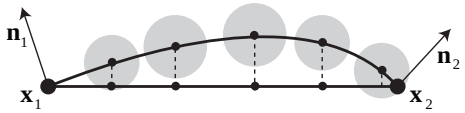

Figure 4: *Instead of linearly interpolating the attachment positions, we use a Bézier patch to hide the piecewise linear shape of the base mesh.*

4. Runtime Update

At each time step, the attachment point of each wrinkle vertex is computed first. The simplest way is to use linear interpolation:

$$\mathbf{a} = (1 - b_0 - b_1)\mathbf{x}_{i_0} + b_0\mathbf{x}_{i_1} + b_1\mathbf{x}_{i_2}, \qquad (1)$$

where $\mathbf{x}_{i0}, \mathbf{x}_{i1}$ and $\mathbf{x}_{i2}$ are the base mesh vertices of the referenced triangle and b_0, b_1 the corresponding barycentric coordinates. However, as Fig. 5 shows, this piecewise linear interpolation yields artifacts when the triangles of the base mesh are substantially larger than the wrinkle mesh triangles. To fix this problem, we use the interpolation proposed in [VPBM01] for rendering polygonal meshes. They approximate the shape of a triangle with a three-sided cubic Bézier patch using both, the corner positions and the normals (see

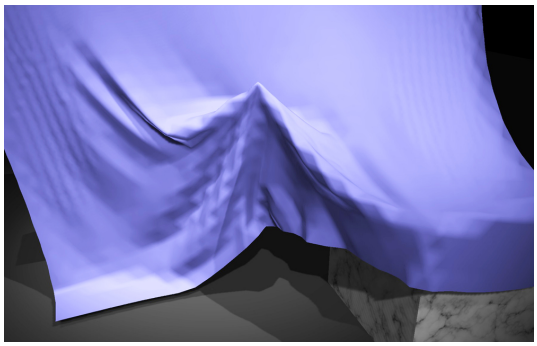

Figure 5: *Attaching the wrinkle vertices to the flat surfaces of the base triangles reveals the piecewise linear shape of the base mesh.*

Fig. 4). This function can be evaluated analytically at any position inside the triangle which is important since the tessellations of the base and the wrinkle mesh are independent. The Bézier patch approximation can still yield small visual artifacts. As future work we plan to use a subdivision scheme to compute the attachment positions of regular wrinkle meshes as proposed by [Tsi06].

The position $\mathbf{p}_i$ of wrinkle vertex i is defined relative to the attachment position as

$$\mathbf{p}_i = \mathbf{a}_i + \mathbf{o}_i, \tag{2}$$

where $\mathbf{o}_i$ is an offset vector. It would be natural to define this offset in the local frame of the base triangle. However, we found that the cheaper way of defining it in world space does not introduce visual artifacts. The offset is stored as a state variable so the static solver can use the solution of the previous time step as a starting point. In other words, we store $\mathbf{o}_i$ instead of $\mathbf{p}_i$. The latter can be computed at any time using Eq. (2).

4.1. Static Solver

We use a simple Gauss-Seidel iterative solver to update the offset vectors such that the constraints on the wrinkle mesh are satisfied. Although converging slower than global solvers in general, the Gauss-Seidel method has several important advantages. It can directly handle over-constrained systems and systems of non-linear and unilateral equations and is easy to implement. In a given iteration, the solver visits each constraint and performs a geometric projection.

For a distance constraint between points $\mathbf{p}_1$ and $\mathbf{p}_2$ with rest length l_0, the positions are updated in a straightforward fashion as

$$\Delta\mathbf{p} = \frac{\mathbf{p}_2 - \mathbf{p}_1}{|\mathbf{p}_2 - \mathbf{p}_1|}(|\mathbf{p}_2 - \mathbf{p}_1| - l_0) \tag{3}$$

$$\mathbf{p}_1 \leftarrow \mathbf{p}_1 + \frac{1}{2}\Delta\mathbf{p} \tag{4}$$

$$\mathbf{p}_2 \leftarrow \mathbf{p}_2 - \frac{1}{2}\Delta\mathbf{p} \tag{5}$$

$$\tag{6}$$

The same projection is applied for bending constraints. However, in contrast to the distance constraints, $\Delta\mathbf{p}$ is scaled with a scalar $0 \leq k_{\text{bending}} \leq 1$ to be able to control the bending stiffness of the wrinkle mesh. Attachment constraints are projected via

$$\mathbf{p} \leftarrow \mathbf{p} + \frac{\mathbf{a} - \mathbf{p}}{|\mathbf{a} - \mathbf{p}|}(|\mathbf{a} - \mathbf{p}| - r) \tag{7}$$

but only if $|\mathbf{a} - \mathbf{p}| > r$, since attachment constraints are unilateral.

After the solve, the offsets are updated as $\mathbf{o}_i = \mathbf{p}_i - \mathbf{a}_i$.

4.2. Collision Handling

We distinguish two types of collision, self-collision and collision with the environment. To reduce self-penetration of the wrinkle mesh we choose the r_i to be smaller than halve the average edge length of the wrinkle mesh (see Section 3.1). The choice cannot give a guarantee that no self collisions occur. Finding a way to prevent self collisions by construction is part of our future work. The visual artifacts of such small scale self collisions are minimal in our examples though if the mesh is drawn two sided.

The collision volumes of purely animated characters as in Fig. 8 are typically much coarser than the graphical mesh and are inflated. To prevent collisions of the environment with the wrinkle mesh, one simply has to make sure that the inflation is larger than r_i.

In the case of clothing simulation, we rely on the cloth simulator to take care of collisions against the character. Typically, outward normals of the base mesh are available. Collisions of the wrinkle mesh can then be avoided by projecting wrinkle vertices to the outer side of the base mesh. Since each wrinkle vertex has a link to the closest base triangle this can be done efficiently by looking at a local neighborhood of base triangles. We take an even simpler approach. We interpolate the outward normals of the base mesh at the attachment points and make sure that the wrinkle vertices stay on the proper side of the plane defined by the attachment point and the interpolated normal. There is yet another approach: Typical cloth simulators give the cloth a finite thickness to increase stability. If this is the case one can simply choose the thickness to be larger than the $2r_i$.

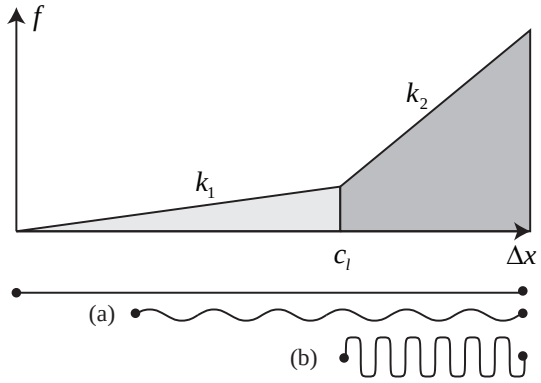

Figure 6: *To model the behavior of the wrinkle mesh under compression, two stiffness constants k_1 and k_2 are assigned to the edges of the base mesh. Up to the compression limit c_l the repulsive force is weak (a). After that point the wrinkle mesh is fully folded up (b) resulting in a higher stiffness k_2.*

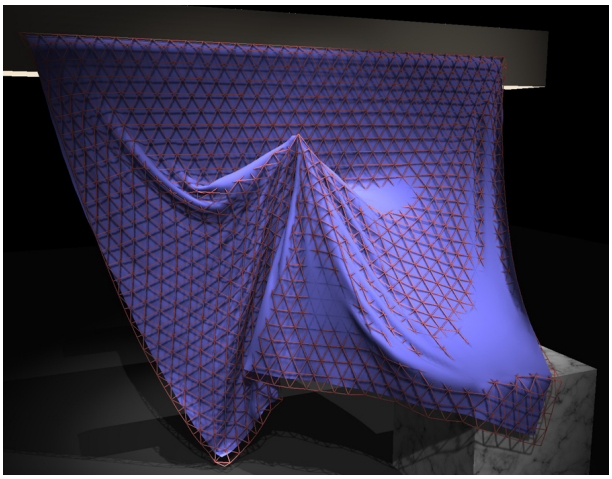

Figure 7: *With reduced compression stiffness the base mesh does not buckle. Instead, its triangles get compressed creating the correct wrinkles in the fine mesh.*

4.3. Weakening Compression Resistance

On animated meshes, edge lengths typically vary enough over time to induce the desired wrinkles. However, in the case of simulated cloth, the simulator that operates on the base mesh makes sure edges do not get compressed. This works against wrinkle formation and makes the mesh move in a way which is not correct for the superimposed fine mesh.

To fix this problem we change the force profile on the cloth edges as shown in figure Fig. 6. Under compression, the spring stiffness is small up to a point defined by the global *compression limit* parameter $0 \leq c_l \leq 1$. If the edge is compressed beyond this point, the stiffness gets larger modeling the effect that the cloth is now fully folded. This modification makes a big difference. Without reducing the compression resistance of the base mesh, the wrinkle mesh looks as shown in the left image of Fig. 1 instead of showing the detail depicted in the right image. We were surprised how well the two step profile actually works. Instead of simply adding wrinkles to a coarse mesh, the wrinkle mesh behaves almost exactly as if it were simulated in full detail. As the accompanying video shows, it is hard to see that the cloth simulation is actually performed on a much coarser mesh than what the user sees on the screen (see Fig. 7).

5. Results

All our examples were run on a single core of an Intel Core2 CPU at 2.4 GHz and an NVIDIA Quadro GPU. We tested our method on animated characters, regular cloth patches and clothing.

The base mesh shown on the left of Fig. 1 and in Fig. 7 contains 2K triangles. We subdivided it to get a wrinkle mesh with 32K triangles. The wrinkle mesh solver takes only 5 ms per time step including attachments computations, 10 iterations over the constraints and normal computation for rendering. This means that the speed of the scene is bound only by the cloth simulator and the renderer.

Fig. 8 shows the base model of a character with 2K triangles. We created a wrinkle mesh for it containing 32K triangles. Due to the same triangle count, the timings are exactly as in the previous scene since the wrinkle solver is not aware of whether the input comes from a cloth simulator or character animation. Allowing the wrinkle vertices to displace in the tangential direction improves the qualities of the wrinkles substantially as the four screenshots on the bottom right show. Wrinkles can form freely across the tesselation of the base mesh and form sharper contours.

We compared our method with example-based wrinkle synthesis [WHRO10] as shown in Fig. 9. In general, wrinkle synthesis yields more realistic results but it is also substantially more expensive then our method in terms of compute time and memory consumption. The top row shows the base mesh, wrinkle synthesis and our method for the same pose. In the middle we show how a self intersecting base mesh is handled in both cases. Wrinkle synthesis hides the situation better than our method but it produces self intersections in the fine mesh which are clearly visible. The bottom row shows the synthesized mesh, our method and the result of our method when painting the pre-stretch factors s explicitly to remove wrinkles in the chest region.

Fig. 10 shows the application of wrinkle meshes to clothing on characters. The base mesh of the skirt has 7K triangles for which we created a wrinkle mesh composed of 28K triangles. The static solver takes 4 ms for 5 iterations per time step in this sample. In the video and just as a technical test, we subdivided the wrinkle mesh further to get 112K

Figure 8: *Left: Base mesh vs. wrinkle mesh. Right top: Normal dof only. Right bottom: All dofs. (Model by Simutronics)*

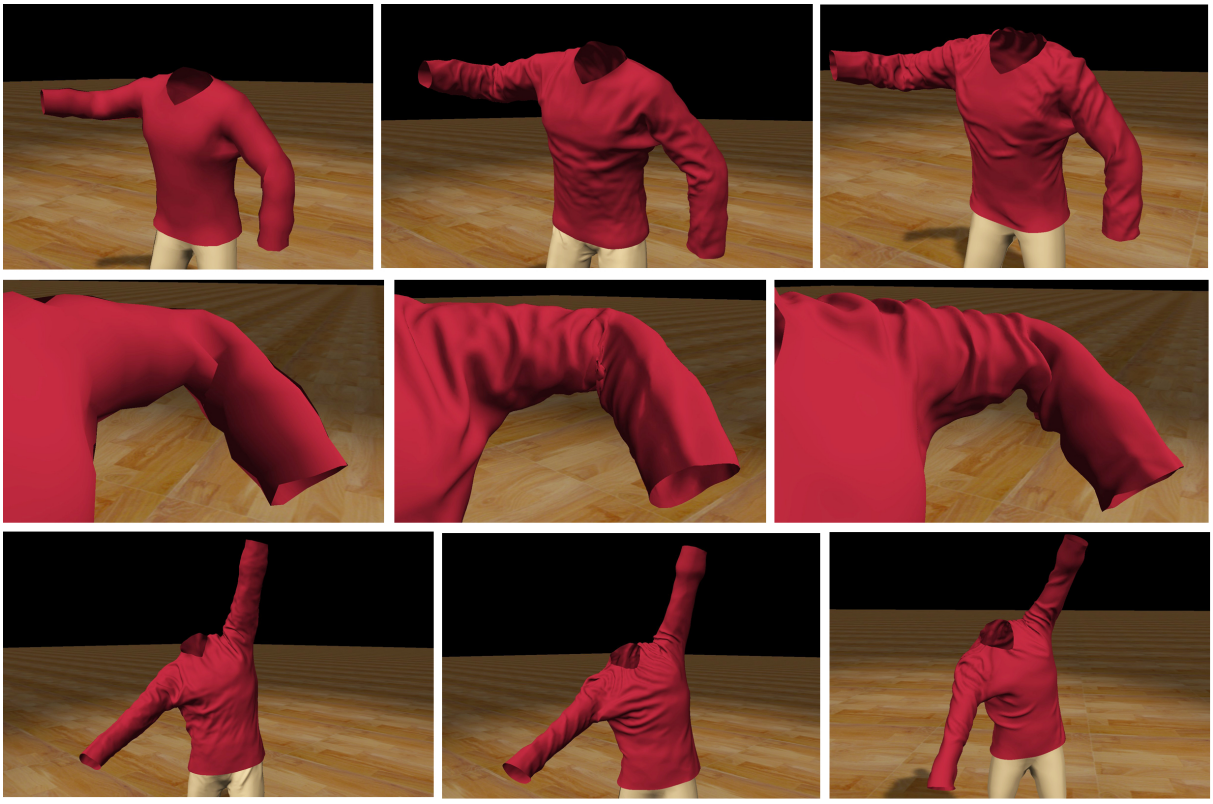

Figure 9: *Comparison of our method with example-based wrinkle synthesis [WHRO10]. Top and middle row: Base mesh, wrinkle synthesis and our method. The middle row shows how self intersection in the base model is handled by the two approaches. Bottom row: Result generated with wrinkle synthesis, our method and our method with painted pre-stretch factors s to reduce the wrinkles in the chest region. Model courtesy of Huamin Wang*

triangles even though the visual result becomes less plausible. Even in this extreme case, the cloth solver and renderer remain the bottlenecks since the wrinkle mesh solver only takes 10 ms per time step.

6. Conclusion and Future Work

We presented a simple, yet effective method for adding wrinkles to animated characters or simulated cloth. The fact that wrinkle generation is fully automatic allows the generation of convincing high resolution cloth behavior on top of coarse simulation meshes.

In the case of animated characters, more control that we can currently provide would be desirable. One of the main drawbacks of our method is that there is no simple way to control or specify the directions of the wrinkles since they emerge automatically in the expected directions. In the future we plan to extend our method to give the artist more direct control on wrinkle directions.

In addition to animated skin or simulated cloth, wrinkle meshes are well suited to add detail to soft bodies as well. Soft bodies are typically simulated using tetrahedral meshes. Instead of showing tetrahedra, high resolution surface meshes are used for rendering which are attached to the tetrahedral meshes using barycentric coordinates exactly as in the wrinkle mesh case. Therefore, a few modifications will allow us to add wrinkles to soft bodies as well.

Currently, we are in the process of including our solver into a commercial game library.

References

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 28–36.

[BRW*07] BICKEL B., ROL M. B., WOJCIECH A., MIGUEL M., PFISTER O. H., GROSS M.: Multi-scale capture of facial geometry and motion. In *ACM Trans. on Graphics* (2007), vol. 26.

[EFP09] EIBNER G., FUHRMANN A., PURGATHOFER W.: Generating predictable and convincing folds for leather seat design. In *Proceedings of the 25th Spring Conference on Computer Graphics (SCCG)* (2009), pp. 93–96.

[HBVT99] HADAP S., BANGERTER E., VOLINO P., THALMANN N. M.: Animating wrinkles on clothes. In *VIS '99: Proceedings of the conference on Visualization '99* (1999), pp. 175–182.

[Jak01] JAKOBSEN T.: Advanced character physics in the fysix engine. *www.gamasutra.com* (2001).

[KC02] KANG Y.-M., CHO H.-G.: Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. *Computer Animation* (2002), 203.

[KWH04] KIMMERLE S., WACKER M., HOLZER C.: Multilayered wrinkle textures from strain. In *VMV* (2004), pp. 225–232.

[LC04] LARBOULETTE C., CANI M.-P.: Real-time dynamic wrinkles. In *Computer Graphics International 2004* (June 2004), pp. 522–525.

[Lov06] LOVISCACH J.: Wrinkling coarse meshes on the gpu. *Comput. Graph. Forum 25*, 3 (2006), 467–476.

[MHR06] MÜLLER M., HENNIX B. H. M., RATCLIFF J.: Position based dynamics. *Proceedings of Virtual Reality Interactions and Physical Simulations* (2006), 71–80.

[MTG04] MÜLLER M., TESCHNER M., GROSS M.: Physically-based simulation of objects represented by surface meshes. In *Proceedings of Graphics Interface (GI 2004)* (2004), pp. 26–33.

[MTV05] MAGNENAT-THALMANN N., VOLINO P.: From early draping to haute couture models: 20 years of research. *The Visual Computer 21*, 8-10 (2005), 506–519.

[Mül08] MÜLLER M.: Hierarchical position based dynamics. *Proceedings of Virtual Reality Interactions and Physical Simulations* (2008).

[NMK*06] NEALEN A., MÜLLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (2006), 809–836.

[RMB08] REIS C. D. G., MARTINO J. M. D., BATAGELO H. C.: Real-time simulation of wrinkles. In *Proceedings of WSCG* (2008).

[Tsi06] TSIKNIS D.: *Better Cloth Through Unbiased Strain Limiting and Physics-Aware Subdivision*. Master's thesis, The University Of British Columbia, 2006.

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), ACM, pp. 159–166.

[WHRO10] WANG H., HECHT F., RAMAMOORTHI R., O'BRIEN J.: Example-based wrinkle synthesis for clothing animation. In *Proceedings of ACM SIGGRAPH* (2010).

[WWY06] WANG Y., WANG C. C. L., YUEN M. M.-F.: Fast energy-based surface wrinkle modeling. *Computers & Graphics 30*, 1 (2006), 111–125.

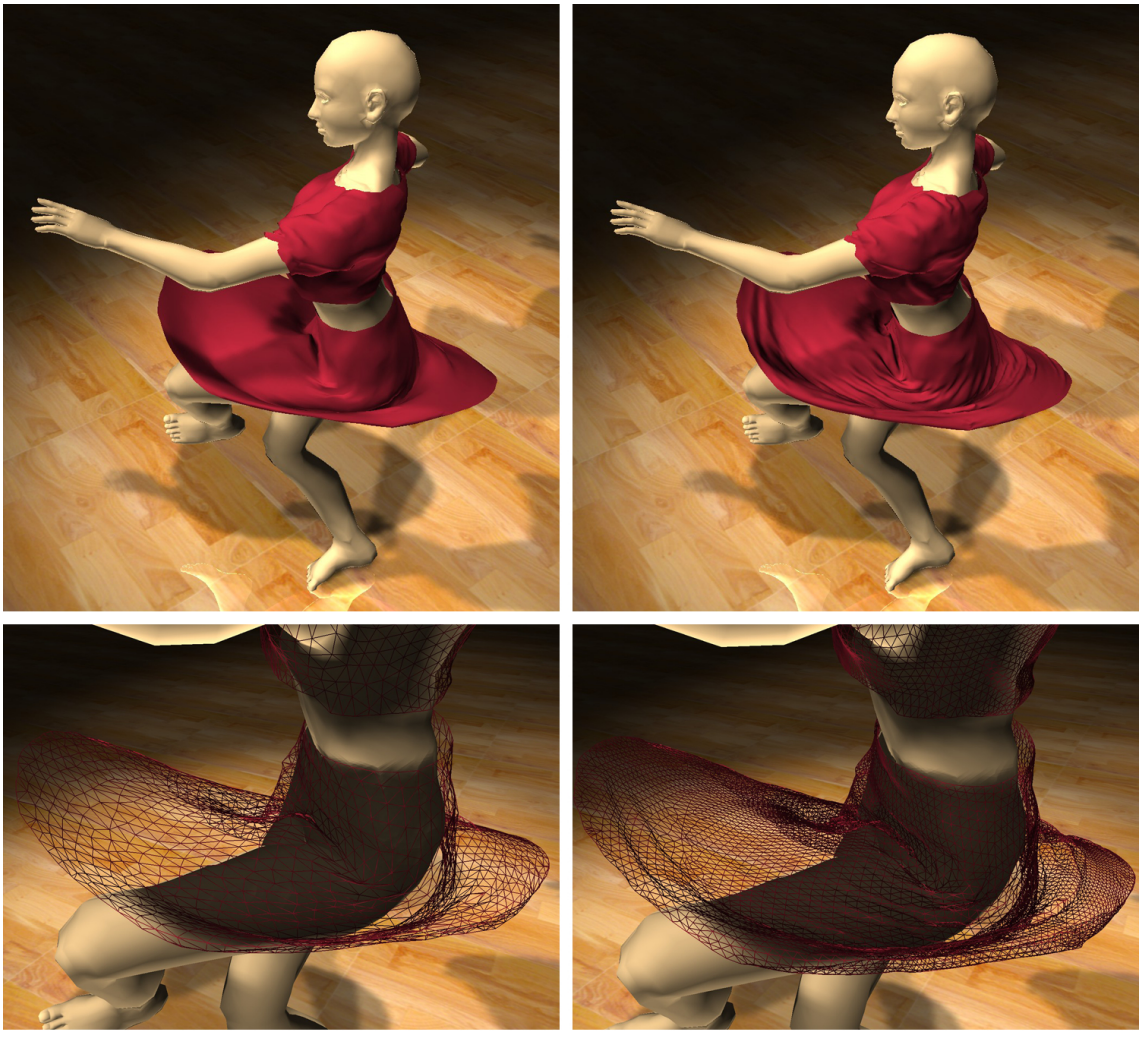

Figure 10: *Left: Base mesh with 7K triangles. Right: Wrinkle mesh comprised of 28K triangles.*

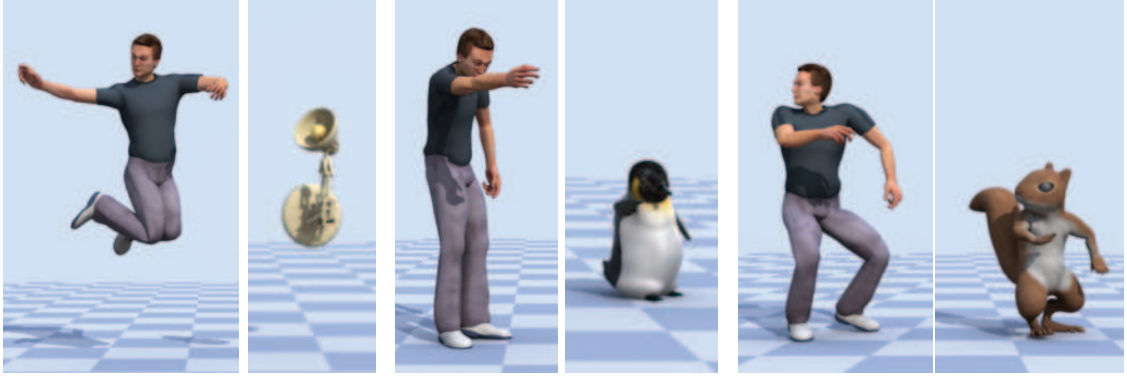

Figure 10: *Non-humanoid characters animated using human motion capture data.*