Multi-Resolution Cloth Simulation

Yongjoon Lee 1  
Sung-eui Yoon 1  
Seungwoo Oh 2  
Dyksu Kim 1  
Sunghee Choi 1

1 KAIST (Korea Advanced Institute of Science and Technology)  
2 CLO Virtual Fashion Inc.

Abstract
We propose a novel, multi-resolution method to efficiently perform large-scale cloth simulation. Our cloth simulation method is based on a triangle-based energy model constructed from a cloth mesh. We identify that solutions of the linear system of cloth simulation are smooth in certain regions of the cloth mesh and solve the linear system on those regions in a reduced solution space. Then we reconstruct the original solutions by performing a simple interpolation from solutions computed in the reduced space. In order to identify regions where solutions are smooth, we propose simplification metrics that consider stretching, shear, and bending forces, as well as geometric collisions. Our multi-resolution method can be applied to many existing cloth simulation methods, since our method works on a general linear system. In order to demonstrate benefits of our method, we apply our method into four large-scale cloth benchmarks that consist of tens or hundreds of thousands of triangles. Because of the reduced computations, we achieve a performance improvement by a factor of up to one order of magnitude, with a little loss of simulation quality.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction
Cloth simulation has been extensively researched in order to achieve realistic simulations of various fabrics. Since most fabrics are very flexible and do not have elasticity, the meshes resulted from realistic cloth simulations can have highly detailed features such as wrinkles. Therefore, the resolutions of meshes used for high-quality cloth simulations are typically very high enough to capture such detailed features. However, using such high resolution meshes can cause significantly slow simulation performances, especially since the time complexity of most high-quality cloth simulations is higher than linear functions with the number of vertices of the mesh [GHF*07].

Many orthogonal approaches have been proposed to accelerate the performance of cloth simulation. At a high level, they include allowing larger time steps [BW98], GPU-based parallelization [Gre03], wrinkle synthesis [WHRO10], faster collision detection [KHH*09,GKJ*05], and multi-resolution approaches [VB05].

Existing multi-resolution techniques [VB05, HPH96] achieve a higher simulation performance by providing adaptive meshes. These techniques identify mesh regions that require high accuracy and use more high resolutions only for those regions instead of using a uniformly refined mesh. However, existing multi-resolution techniques do not attempt to simplify mesh regions where can be represented

Figure 1: The left figure shows simulation of a dress in a woman, who is not shown in this figure. The right figure shows the multi-resolution mesh used to perform the simulation at a particular frame. The original dress mesh has 25 K vertices. By simplifying dynamically smooth regions of the dress mesh, our method achieves 9 times performance improvement by reducing 73.8% of the vertices of the original mesh.
with lower resolutions while providing plausible simulation quality. Moreover, these techniques have been designed only for mass-spring models and are not directly applicable to more general cloth models [BW98, CK02, GHF 07] that are widely used for high-quality cloth simulation.

Main contributions: In this paper we propose a novel multi-resolution cloth simulation method that simplifies linear system that do not require a high resolution, in order to improve the performance of cloth simulation while maintaining the simulation quality. We design our multi-resolution approach for a triangle-based energy model with implicit Euler integration. We identify that solutions of the linear system of cloth simulations are smooth in certain regions and solve the linear system in a reduced solution space. Then we construct the original solutions by performing a simple interpolation. In order to identify regions whose solutions for the linear system are smooth, we use simplification metrics that consider various forces and geometric collisions. We have implemented our method and applied it to various cloth benchmarks consisting of up to 100 K vertices. With a little loss of simulation quality, our method achieves the performance improvement by a factor of up to one order of magnitude with our tested benchmarks.

2. Related Work

In this section we review prior work on cloth simulation and its various acceleration techniques.

2.1. Cloth Simulation

The cloth simulation has been extensively researched and good surveys [CK05] are available. Since Breen et al. [BHW94] proposed a particle based cloth model, the mass-spring model has been widely used for efficient and realistic cloth simulation. Provot [Pro96] introduced a simple mass-spring model and an explicit integration-based solver. Although this method runs interactively, it suffers from unstable behaviors as the time step becomes large. For a robust cloth simulation even with large time steps, Baraff and Witkin [BW98] developed an implicit integration method. Choi and Ko [CK02] improved the stability and quality of simulation by introducing an immediate buckling model. Also, Bridson et al. [BFA02] proposed a robust method that handles various contacts. Goldenthal et al. [GHF 07] addressed the over-stretched problem of cloth based on constrained Lagrangian mechanics to constrain the distance between particles. Wang et al. [WHRG10] recently proposed an example-based wrinkle synthesis technique, which combines fine wrinkles with coarse cloth simulation. Aguier et al. [dA STH10] presented a learning-based approach to model the dynamic behavior of clothes and used it for efficient cloth simulation.

2.2. Adaptive Cloth Simulation

Many adaptive techniques have been proposed to improve the performance and quality of cloth simulations. Hutchinson et al. [HPBH96] and Zhang et al. [ZY01] proposed adaptive cloth simulation methods based on the mass-spring model. These methods treat the mass-spring model of a cloth as a mesh and refine a portion of the mesh that requires a higher mesh resolution. However, these methods showed much lower simulation quality compared to using a uniform mesh that has a higher resolution. Villard et al. [VB05] also proposed an adaptive meshing method. This method preserved the force momentum during the refinement of the adaptive mesh and thus achieved a simulation quality similar to that of the original cloth simulation that use uniform meshes. However, all these prior adaptive techniques are based on the mass-spring model and are not easily applicable to other types of cloth models (e.g., triangle-based energy models [BW98]) that can show high cloth simulation quality.

2.3. Adaptive Techniques in Other Fields

Adaptive techniques have been extensively studied in various simulations. Grinspun et al. [GKS02] addressed adaptive simulation for finite element methods by refinement of basis functions. An et al. [AKJ08] use optimized cubatures to reduce the size of various linear equation systems. Their method requires a training set to compute optimized cubatures.

Losasso et al. [LG F04] use an octree data structure to simulate water and smoke efficiently. They discretize the linear system on an unrestricted octree grid so that they reduce the size of the linear system. Agarwala [Aga07] apply this idea to constructing a seamless large-scale image panorama that requires solving the Poisson equation. He reduces the size of the linear system drastically, since the solutions for the Poisson equations are smooth. Our method is inspired by this adaptive method that reduces the size of the linear system by merging smooth solutions into one.

2.4. Multigrid Methods

Multigrid methods have been widely used in many different simulations. Its main idea is to accelerate the convergence of a basic iterative method, accomplished by solving a problem with regularly, coarser domains in multiple levels. However, these multigrid methods have not been actively employed in cloth simulation, mainly because the simulated cloth meshes have many fine details such as wrinkles. Müller [Müi08] simulated clothes with a hierarchical multigrid solver for a position-based dynamics. Oh et al. [ONW08] applied the physically faithful standard v-cycle multigrid to cloth simulation. Our method shows more efficiency than this standard multigrid method for cloth simulation.

3. Overview

In this section, we give an overview of our approach to efficiently simulate large-scale clothes.

3.1. Issues of Large-Scale Cloth Simulations

Cloth is highly constrained because of inextensibility and collision constraint. The reality of cloth simulation depends
We assume in this paper that cloth simulation method uses a triangle-based energy model constructed from a cloth mesh, integrated with the well-known implicit Euler integration method [BW98]. However, our method can be easily extended to other types (e.g., mass-spring model) of energy models and other integration methods, if one can design a linear system from those cloth models and integration methods.

The implicit Euler integration for the triangle-based energy model from a cloth mesh gives the following linear equations [BW98]:

\[ ( \mathbf{M} - h \left( \frac{\partial \mathbf{f}}{\partial \mathbf{v}} + h \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right) ) \Delta \mathbf{v} = h \left( \mathbf{f} + h \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{v} \right), \] (1)

where \( h \) is a length of time step, \( \mathbf{M} \) is a mass matrix, \( \mathbf{f} \) is a force vector, \( \mathbf{x} \) and \( \mathbf{v} \) are position and velocity vectors defined for each vertex of the cloth mesh, and \( \Delta \mathbf{v} \) is a difference between two velocities of \( \mathbf{v}^{t+1/2} \) and \( \mathbf{v}^{t} \).

Eq. (1) can be rewritten as the following standard linear system using the fact that \( \Delta \mathbf{v} = h - \mathbf{v}^{t+1/2}; \)

\[ \mathbf{A} \Delta \mathbf{x} = \mathbf{b} \] (2)

We perform our simplification method in the space of \( \Delta \mathbf{x} \) instead of \( \Delta \mathbf{v} \), since \( \Delta \mathbf{x} \) is likely to be smooth for regions of the cloth mesh, even though \( \Delta \mathbf{v} \) varies for those regions.

4.1. Dynamically Smooth Regions

Even though the cloth mesh can have very complex geometric shapes such as wrinkles, velocities \( \Delta \mathbf{x} \) on portions of the mesh can vary smoothly.

In order to identify mesh regions that can be simplified with our method, we decompose the mesh into the following two regions: dynamically smooth and complex regions. Intuitively speaking, a mesh region belongs to the dynamically smooth regions, if all the vertices of the mesh region have similar velocities \( \Delta \mathbf{x} \).

Fig. 3 shows an example of the cloth mesh with regions that can be classified as dynamically smooth regions and thus can be simplified with our method. In this example, a ball collides with a cloth and pushes it. Fig. 3-(b) shows velocities of each vertex, each of whose \( x, y, \) and \( z \) components is visualized in the red, green, and blue channels respectively. In the example cloth mesh, region 2 that has very smooth geometry and also has very smooth velocities. Also, region 1 that has a wrinkled region, a complex geometry, also has similar velocities. Therefore, region 1 and 2 can be classified as dynamically smooth regions. However, region 3 that has different velocities is classified as the dynamically complex region and thus is not simplified in our method.

\[ \mathbf{v}^{t+1/2} \] is an updated velocity with the information (e.g., velocities, forces and Jacobians) of frame \( t \). \( \mathbf{v}^{t+1/2} \) is then calculated with \( \mathbf{v}^{t+1/2} \) and the collision information of frame \( t \).
Once we identify dynamically smooth regions, we simplify linear equations related to those regions and solve solutions for the simplified equations in a reduced space, \( \Delta y \), instead of the original space \( \Delta x \). Then, \( \Delta x \) that are simplified into \( \Delta y \) can be computed by interpolating elements of \( \Delta y \). For this, we introduce a transformation matrix, \( S \), that transforms a reduced space, \( \Delta y \), to the solution space, \( \Delta x \), of the original linear equations. As a result, Eq. 2 is transformed into the following equation:

\[
A S \Delta y = b,
\]

where \( \Delta x = S \Delta y \), and \( \Delta y \) is a vector of length \( O(n) \), which is much smaller than the length \( O(m) \) of the vector \( \Delta x \).

4.2. Construction of Multi-Resolution Representation

In order to efficiently perform the simplification on the dynamically smooth regions, we utilize a multi-resolution representation built from an input cloth mesh.

We apply the Loop subdivision [Loo78] to the input coarse mesh in order to create a high-resolution mesh for a high-quality cloth simulation. The Loop subdivision of a triangle \( T \) creates its four child triangles, \( T_0, T_1, T_2, \) and \( T_3 \) (Fig. 4). We use \( T_0 \) to denote the center child triangle, and \( T_1, T_2, \) and \( T_3 \) to denote the other child triangles.

Based on this parent-child relationship, we construct a multi-resolution forest whose root nodes correspond to triangles of the input cloth mesh by recursively performing the Loop subdivision a few times (e.g., 3 or 4 times) before running the cloth simulation.

At each runtime simulation step, we maintain a simplification cut that approximates the original high-resolution cloth mesh. The simplification cut is initialized with all the leaf nodes of the multi-resolution hierarchy and thus represents the original high-resolution mesh. Then, as we traverse triangles (i.e., nodes) of the simplification cut, we get four sibling triangles and check whether we can simplify them into their common parent triangle in our multi-resolution hierarchy, according to our simplification metrics that define the dynamically smooth regions. This operation can be efficiently performed by traversing our multi-resolution representation in a bottom-up manner.

4.3. Simplification Metrics

In order to identify dynamically smooth regions, we present simple simplification metrics. Intuitively speaking, as more different forces are applied to a region, it is likely that vertices of the region will have more different accelerations at the current frame. There are three major forces that are important to cloth simulations: shear, stretching, and bending forces. In explicit Euler method, we can easily calculate acceleration for each vertex by calculating \( M^{-1}f \). However, we cannot use these forces directly as a measurement of acceleration in the implicit Euler method; as in Eq. 1, \( \Delta x \) is multiplied by \( A \) that consists of the mass matrix \( M \) and the Jacobian matrix \( \partial f / \partial x \).

Instead, we propose to identify dynamically smooth regions based on geometric information computed at the current and previous frames. Because we perform our simplification method in the space of velocity (\( \Delta x \)), we consider both the velocity of the previous frame and the difference of geometric features between the previous and current frames, as an estimation for the acceleration in the current frame to determine whether a region is dynamically smooth or not. Intuitively speaking, if the differences of geometric features (e.g., dihedral angles and shapes) of triangles of a mesh region between two consequent frames are similar, then we can conclude that the triangles of the mesh region had similar accelerations and thus it is also likely that accelerations of those triangles are similar even in this frame assuming the frame-to-frame temporal coherence. More specifically, we check velocities of triangles to find whether they have similar velocities, followed by checking geometric differences caused by shear, stretch, and bending forces.

**Velocity** Suppose that we consider four triangles, \( T_0, T_1, T_2, \) and \( T_3, \) in the simplification cut, to see whether we can simplify them for their parent triangle \( T \) shown in Fig. 4. For each edge of \( T, \) we compare the velocity vector of its midpoint (e.g., \( v_3 \) in Fig. 4) with the average velocity vector from the velocity vectors of its two endpoints (e.g., \( v_0 \) and \( v_1 \) in...
Fig. 4. If these two velocities are similar in terms of the magnitude and direction of these two velocities, we further check in-plane and out-of-plane deformations for the simplification. Otherwise, we do not simplify the four triangles. Then, we fetch other four triangles in the simplification cut and continue to check them for the simplification.

In-plane deformation In-plane deformation on cloth meshes is governed by stretch and shear forces. If strong in-plane forces are exerted on a triangle, those forces deform the triangle along the in-plane direction. For estimating stretch and shear forces, we check the difference of stretch factor \((w_{u}, w_{v})\) and shear factor \((w_{u+v}, w_{u-v})\) during one time step \([BW98]\). We can calculate \(w_{u}\) and \(w_{v}\) of triangle \(T_{ijk}\) consisting of vertices \(i, j, k\) as

\[
(w_{u}, w_{v}) = (x_{j} - x_{i}, x_{k} - x_{i}) \cdot (u_{j} - u_{i}, u_{k} - u_{i})^{-1},
\]

where \(x_{i}\) is the position of vertex \(i\) in the world coordinate, \(u_{i}\) is the rest position of vertex \(i\) in the 2D-domain coordinate. We can calculate \(w_{u+v}, w_{u-v}\) by \(w_{u+v} = w_{u} + w_{v}\), and \(w_{u-v} = w_{u} - w_{v}\) \([CK02, ONW08]\). We then define our simplification metrics for stretching and shearing at frame \(t\) as follows:

\[
M'_{\text{stretch}}(T) = A(T) \frac{\Delta |w'_{u}| + \Delta |w'_{v}|}{h},
\]

\[
M'_{\text{shear}}(T) = A(T) \frac{\Delta |w'_{u+v}| + \Delta |w'_{u-v}|}{h},
\]

where \(A(T)\) is the area of triangle \(T\) and \(\Delta |w'| = |w'| - |w'|^{-1}\). Since as the area of a triangle becomes larger and the difference becomes larger at a unit time, there are more stretching and shearing forces on the triangle, we consider them in our simplification metrics.

Out-of-plane deformation In order to estimate the difference of the bending force between two neighboring triangles, we consider the difference of the dihedral angle of those two neighboring triangles at the previous and current frames. We use \(\angle T_{j}\) to denote the angle between two triangles \(T_{0}\) and \(T_{j}\) at frame \(t\), shown in Fig. 4. Then the simplification metric for the bending force of a triangle \(T\) at frame \(t\) is defined as the following:

\[
M'_{\text{bend}}(T) = \max_{j=1,2,3} A(T) \frac{\Delta \angle T_{j}}{h},
\]

where \(\Delta \angle T_{j}\) is the difference between \(\angle T_{j}^{-1}\) and \(\angle T_{j}\). The area of a triangle \(A(T)\) is also considered as in the in-plane deformation.

Collisions Our simplification metrics for shear, stretch, and bending forces are based on the frame-to-frame temporal coherence. However, once we have self-collisions within the cloth mesh or inter-collisions between the cloth mesh and other objects, the frame-to-frame temporal coherence is not satisfied. Also, the frame-to-frame coherence breaks even when collisions among meshes are resolved. Therefore, we do not simplify vertices or triangles at the moment when their collision states are changed.

4.4. Computing Simplified Solutions

We first construct the transformation matrix \(S\) according to our simplification metrics, by traversing our multi-resolution hierarchy in a bottom-up manner. Then we compute solutions of the linear system defined in a reduced solution space.

Formulating the transformation matrix \(S\) We traverse our multi-resolution hierarchy in a bottom-up manner to construct the simplification cut that represents the simplified cloth mesh at the current frame. Once we compute the simplification cut, then we can formulate the transformation matrix \(S\). The original solutions in the space of \(\Delta x\) can be represented by a weighted sum of simplified solutions computed in the space of \(\Delta x\). For example, suppose that we simplify four sibling triangles, \(T_{0}, T_{1}, T_{2}\), and \(T_{3}\), shown in Fig. 4 into their parent triangle \(T\). Once we compute solutions for \(v_{0}\), \(v_{1}\), and \(v_{2}\) for the parent triangle \(T\), then the solution of \(v_{3}\) can be computed by the average of solutions of \(v_{0}\) and \(v_{1}\); solutions of other vertices \(v_{4}\) and \(v_{5}\) can be computed in a similar manner.

Solving the reduced linear system In order to solve the linear system of Eq. \(3\) that is defined in the reduced solution space, we multiply \(S^{T}\) to both sides of the equation, resulting in the following equation:

\[
S^{T} AS \Delta y = S^{T} b. \tag{4}
\]

The size of \(S^{T} AS\) becomes \(3m \times 3m\), where \(m\) is the number of vertices of the simplified cloth mesh. We use PCG to solve Eq. 4. Computing \(S^{T} AS\) can be done quickly, since \(A\) and \(S\) are sparse. For a cloth that consists of 20 K triangles, computing \(S^{T} AS\) takes only 10 ms to 15 ms, while solving the original linear system takes 500 ms to 1000 ms.

4.5. Error Correction

We approximate the original solution, \(\Delta x_{\text{sol}}\), of Eq. \(2\), as the product of \(S\) and the solution, \(\Delta y_{\text{sol}}\) of Eq. \(4\). However, \(S \Delta y_{\text{sol}}\) is not exactly the same as \(\Delta x_{\text{sol}}\), resulting in error, \(e\).
Especially, it is likely that the error becomes bigger when we have collisions, which breaks the frame-to-frame coherence.

The error $e$ consists of high-frequency errors (e.g., bumps and unrealistic wrinkles) and low-frequency errors (e.g., the overall shape of cloth). Artifacts caused by high-frequency errors look more unpleasant than those caused by low-frequency errors.

In order to reduce errors, especially visually unpleasant high-frequency errors, we perform error correction with the residual equation. We formulate the residual equation of Eq. 2 as the following:

$$Ae = b - AS\Delta y.$$  

As we perform iterations with PCG on Eq. 5, high-frequency errors drastically decrease with only a few iterations [Wes92]. Fig. 5-(a) shows the mean and variance of errors on a frame of one of our benchmarks. Note that the variance of errors are drastically reduced by only a few iterations.

5. Results

We have implemented our method on a PC with a 3.0GHz Intel CPU, an NVIDIA GeForce 8800 GTX, and 2GB main memory. For collision detection, we use a hybrid parallel continuous collision detection method [KHH+09]. To handle collisions, we use the velocity filtering with the repulsion force method as proposed by Bridson et al.'s [BFA02].

We test our method with four different benchmarks (Fig. 6). Our two benchmarks are simple cases: draping a rectangular cloth with two fixed points (Fig. 6-(a)), a cloth draping on a ball (Fig. 6-(b)), a walking woman in a dress (Fig. 6-(c)), and a walking man in trousers (Fig. 6-(d)).

In order to define dynamically smooth regions, we use the following thresholds. If the magnitudes of two velocity vectors are within 10% difference and their angle is less than $0.056\pi$, we consider them to be similar. Also, we use $0.001m^2/s$ as a threshold for $M_{\text{stretch}}$ and $M_{\text{shear}}$, and $0.0013\pi m^2/s$ as a threshold for $M_{\text{bend}}$. We perform 5 to 10 iterations for the error correction; we use 5 iterations for subdivision level 3, and use 10 iterations for subdivision level 4.

We use 0.01s time step size for simulating benchmarks shown in Fig. 6-(a) and 6-(b). However, we use 0.001s time step size for simulating benchmarks shown in Fig. 6-(c) and 6-(d), since these two benchmarks have complex contacts between the cloth meshes and walking characters. Initial cloth meshes in these benchmarks are designed in low resolutions and thus are inappropriate for high-quality cloth simulation. Therefore, we perform the Loop subdivision recursively 3 or 4 times to the base mesh. Also, during the refinement process, we build our multi-resolution hierarchy as mentioned in Sec. 4.2.

Table 1 shows the average simulation time (excluding time spent on collision detection and handling) of our method and PCG with tested benchmarks. Overall our method achieves 8 times on average with all the tested benchmarks and up to 13 times performance improvement over using PCG with the original linear system. Also, as the complexity of cloth meshes increases, we observe that our method shows higher performance improvements over PCG. Although our method approximates the solutions of the original linear equations, we found that there are little noticeable visual artifacts and the simulation results of our method is similar to those computed by PCG.

Our method has four main components: 1) initializing and setting up the linear equation ($\text{Init}$), 2) performing the simplification and constructing the transformation matrix ($\text{Simp}$), 3) performing PCG in the reduced solution space ($\text{Solve}$), and 4) performing the error correction ($\text{EC}$). We measure how much each component takes over the total simulation time. $\text{Solve}$ takes the biggest portion (e.g., 40% to 50%) and other components take similar portions.

We also measure how many triangles of each subdivision level are used in the benchmark of the walking woman with a dress that has the subdivision level of 4 in the original high-resolution mesh. 68% and 22% of triangles of the simplified cloth mesh have subdivision levels of 3 and 4 respectively. Other triangles are drastically simplified to have subdivision levels of 1 and 2.
5.1. Discussion

Time complexity The error correction has $O(kn)$ time complexity [She94], where $n$ is the number of vertices in the original high-resolution mesh and $k$ is the number of iterations for the error correction. Therefore, the time complexity of our overall approach is $O(m^{1.5}+kn)$, where $m$ is the number of vertices in the simplified mesh. In practice, $m$ is much smaller than $n$ (e.g., less than 30% of $n$ for our benchmarks) and we use only 5 to 10 iterations for the error correction. As a result, we are able to observe 8 times performance improvement on average over using PCG to the original linear system.

Comparison with the standard multigrid method The main difference between our method and the standard multigrid is that our method uses adaptive meshes for the simplification according to our simplification metrics, whereas the multigrid uses uniform regular grids with $v$- or $w$-cycles. Such uniform grids may require many iterations for each step of $v$- or $w$-cycles for fine deformations of clothes. Therefore, our method can converge to a solution faster than the multigrid method in a short amount of time. To verify this, we show simulation results (Fig. 7) computed from our method and the standard v-cycle multigrid method [Wes92]. Given the same computation time, our method produces a simulation result (Fig. 7-(b)) that is similar to one computed by solving the original linear equation with PCG (Fig. 7-(a)), while the multigrid method shows very different result (Fig. 7-(c)). To achieve similar simulation quality, the multigrid (Fig. 7-(d)) takes much more time than ours (Fig. 7-(b)).

Limitations We assume that an input high resolution mesh for our method is computed by performing the Loop subdivision to a coarse mesh. We chose this approach, mainly because existing cloth meshes are not designed in high-resolutions and performing the Loop subdivision produces such high quality cloth meshes while facilitating to construct our multi-resolution hierarchy. However, one can simplify an arbitrary input high-resolution mesh by using the well-known edge-collapse simplification operator based on our simplification metrics. We have performed error correction steps with a few iterations (e.g., 5 to 10), in order to mainly reduce the visually unpleasant high-frequency errors in an efficient manner. However, low-frequency errors may remain and thus it is possible to have artifacts on the simulation results.

6. Conclusion

We have designed an efficient multi-resolution cloth simulation technique. We identify dynamically smooth regions and simplify those regions in order to improve the simulation performance without deteriorating the simulation quality. In order to identify such dynamically smooth regions, we have proposed simplification metrics that consider geometric con-

<table>
<thead>
<tr>
<th>Scene</th>
<th>Subd. level</th>
<th>Mat.</th>
<th>No. vertices / No. triangles</th>
<th>Total frames</th>
<th>Ratio (%)</th>
<th>Iter.</th>
<th>Average time of our method (ms)</th>
<th>PCG (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 6-(a)</td>
<td>3</td>
<td>silk</td>
<td>3k/13k</td>
<td>300</td>
<td>11.61</td>
<td>5</td>
<td>16.8</td>
<td>14.3</td>
</tr>
<tr>
<td>Figure 6-(b)</td>
<td>4</td>
<td>silk</td>
<td>2k/25k</td>
<td>300</td>
<td>3.73</td>
<td>10</td>
<td>69.3</td>
<td>59.0</td>
</tr>
<tr>
<td>Figure 6-(c)</td>
<td>3</td>
<td>cotton</td>
<td>4k/22k</td>
<td>300</td>
<td>23.04</td>
<td>5</td>
<td>17.0</td>
<td>14.5</td>
</tr>
<tr>
<td>Figure 6-(d)</td>
<td>4</td>
<td>cotton</td>
<td>2k/61k</td>
<td>300</td>
<td>12.53</td>
<td>10</td>
<td>90.4</td>
<td>58.9</td>
</tr>
<tr>
<td>Figure 7-(a)</td>
<td>3</td>
<td>silk</td>
<td>6k/122k</td>
<td>6000</td>
<td>11.92</td>
<td>5</td>
<td>356.8</td>
<td>282.9</td>
</tr>
<tr>
<td>Figure 7-(b)</td>
<td>3</td>
<td>leather</td>
<td>62k/122k</td>
<td>6000</td>
<td>16.05</td>
<td>5</td>
<td>119.4</td>
<td>97.3</td>
</tr>
</tbody>
</table>

Table 1: This table shows the model complexity, material type, the subdivision level to compute the highest resolution, the average simplification ratio in terms of the number of vertices, and the total number of frames for each tested benchmark. We also show the average simulation time for our method and PCG applied to the original linear system. **Init** – Initializing and setting up the linear system of Eq. 2; **Simp** – Performing the simplification and constructing the transformation matrix; **Solve** – Solving Eq. 4 with PCG; **EC** – Performing the error correction. **Iter** is the number of iterations for the error correction, **PCG** is time spent on performing PCG to the original linear system of Eq. 2.
tacts as well as various forces that govern energy-based cloth models. As a result, we were able to observe 8 times performance improvement on average with the tested benchmarks over running the PCG to the original linear system.

There are many avenues for future research directions. In our current method, we have only designed multi-resolution approach for solving the linear system of cloth simulation. We believe that we can further improve the performance of the cloth simulation by applying multi-resolution collision detection method [YSLM04] without generating unpleasant visual artifacts. Also, we would like to improve the performance of our method by utilizing parallel many-core CPUs and GPUs. Since the main bottleneck of our method is on performing PCG with the linear system even in the reduced solution space, GPU-based parallel solvers [BFGS03] can further improve the performance of our method. Currently, we do not utilize any temporal coherences when we compute the simplified solutions. We believe that we can further improve the performance of our method by utilizing temporal coherences in a similar manner used in progressive deforming meshes [HCC06]. Finally, we would like to support tearing within our method. Since tearing of clothes requires high-resolution on the tearing boundary, we believe that our multi-resolution framework is very promising to support such effects.

Acknowledgment

We would like to thank anonymous reviewers for their constructive feedbacks. This project was supported in part by MKE/MCST/ITA [2008-F-033-02], MKE/ITA u-Learning, MKE digital mask control, MCST/KOCCA-CTR&DPR-2009, KRF-2008-313-D00022, KMCC, BK, DAPA/ADD (UD080042AD), MKE/IITA u-Learning, MKE digital mask control, their constructive feedbacks. This project was supported such effects. That our multi-resolution framework is very promising to improve the performance of our method by utilizing temporal coherences in a similar manner used in progressive deforming meshes [HCC06]. Finally, we would like to support tearing within our method. Since tearing of clothes requires high-resolution on the tearing boundary, we believe that our multi-resolution framework is very promising to support such effects.

References


[Pro96] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In Graphics Interface (1996), pp. 147–154. 2


