View-Dependent Adaptive Cloth Simulation

Journal:	*Transactions on Visualization and Computer Graphics*
Manuscript ID:	Draft
Manuscript Type:	Regular
Keywords:	I.3.5.i Physically based modeling < I.3.5 Computational Geometry and Object Modeling < I.3 Computer Graphics < I Computing Methodologies, I.3.7.a Animation < I.3.7 Three-Dimensional Graphics and Realism < I.3 Computer Graphics < I Computing Methodologies, I.6.8.a Animation < I.6.8 Types of Simulation < I.6 Simulation, Modeling, and Visualization < I Computing Methodologies

View-Dependent Adaptive Cloth Simulation

Woojong Koh, Rahul Narain, and James F. O'Brien

Abstract—This paper describes a method for view-dependent cloth simulation using dynamically adaptive mesh refinement and coarsening. Given a prescribed camera motion, the method adjusts the criteria controlling refinement to account for visibility and apparent size in the camera's view. Objectionable dynamic artifacts are avoided by anticipative refinement and smoothed coarsening. This approach preserves the appearance of detailed cloth throughout the animation while avoiding the wasted effort of simulating details that would not be discernible to the viewer. The computational savings realized by this method increase as scene complexity grows, producing a $2\times$ speed-up for a single character and more than $4\times$ for a small group.

Index Terms—Physically based modeling, animation

1 Introduction

Cloth simulation for visual effects has reached a mature state where the use of virtual characters wearing simulated clothing is now widespread. However, cloth simulation remains computationally expensive, particularly when films require high-quality, realistic results computed at high resolution. For characters that are far from the camera, or otherwise less visible in a shot, fine details will not be visible to the viewer and work spent computing those details is wasted. In most film production settings, both the camera and character motions are known before the simulations are run, and one could use this information to substitute cheaper low-resolution simulations on distant or out-of-frame characters. Unfortunately, manually swapping some characters to low-resolution simulations is cumbersome, particularly when a single character's clothing requires high resolution for some parts of a shot but not others, yet the cloth motion must nevertheless appear coherent throughout. For closeup shots where only part of a character is in frame, savings could also be realized by reducing the computation devoted to out-of-frame cloth, so long as such savings do not result in dynamic artifacts that affect visible parts of the cloth.

In this paper we describe a method for *view-dependent simulation* using dynamically adaptive mesh refinement and coarsening. Instead of using a fixed-resolution simulation mesh, the mesh undergoes local adaptation based on the geometric and dynamic detail of the simulated cloth. The degree to which this detail is resolved is adjusted locally based on the view of the simulated cloth. Areas that appear large in the camera will be refined to show finely detailed dynamics. Areas that are out of frame, facing away from the camera, or at a distance will be correspondingly coarser.

The goal of this work is to preserve the appearance of detailed simulation throughout the animation while avoiding the wasted effort of simulating details that will not be apparent to the viewer. Further, there should be no visible dynamic artifacts created due to varying refinement as the camera and objects move about. Finally, cloth that leaves and reenters visibility should appear to have coherent and consistent dynamics.

Fig. 1. The clothing on this character uses view-dependent simulation. Visible areas (blue) are simulated at high resolution, as measured in screen-space. Areas not visible to the camera (red) are simulated at reduced resolution. The first two images show views from the camera's perspective. The rightmost image shows an outside perspective with the camera's view frustum drawn in black wireframe.

Our work builds on the publicly available ARCSim framework (http://graphics.berkeley.edu/resources/ARCSim) which can be used to animate sheets of deformable materials such as cloth, paper, plastic, and metal. ARCSim adaptively refines and coarsens anisotropic triangle meshes to efficiently resolve the geometric and dynamic detail of the simulated objects. Our method modifies the metrics used by ARCSim so that local mesh visibility is accounted for during refinement. Given the existing framework for adaptivity, our view-dependent refinement is easy to implement and has negligible overhead.

In cases where only a single character is being modeled, our approach realizes modest savings of roughly a $2.4\times$ speed-up in comparison with ARCSim's default adaptive simulation. These savings are due to coarsening out-of-view and back-facing parts of the character's clothing. For small crowd scenes, the savings are larger, up to $4.5\times$, as background and out-of-view characters are automatically coarsened. Compared to a non-adaptive simulation, the speed-up is more than $9\times$. For massive crowd scenes with thousands of agents, we expect that even greater savings could be realized with our approach.

• *W. Koh, R. Narain, and J. F. O'Brien are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. E-mail: {wjkoh,narain,job}@berkeley.edu.*

2 Related Work

Adaptive discretizations have been found to give significant performance and scalability benefits for a number of computationally intensive simulation tasks. In fluid simulation, detailed liquid surfaces can be animated efficiently by refining the spatial resolution near the surface using octrees [1], adaptively sampled particles [2], tall-cell grids [3], or tetrahedral meshes [4], [5], [6]. Adaptive refinement and simplification techniques have also been proposed for mass-spring systems [7], articulated bodies [8] and finite element models [9]. Most relevant to our work are techniques for adaptive cloth simulation [10], [11], [12], [13], [14], which use remeshing to resolve detailed wrinkles and folds. The approach of Narain et al. [14] has also been extended to efficiently model plastic deformation and sharp creases [15] as well as complex fracture patterns [16]. However, all those techniques are view-independent and element sizing is controlled only by geometrical and dynamical properties of the simulated system.

For animation applications, a number of techniques have also been proposed that take into account the current viewpoint and attempt to expend less computational effort in regions that are visually less important. One approach, known as simulation level of detail, is to switch between dynamical models of varying degrees of simplification, depending on the viewpoint. Carlson and Hodgins [17] introduced such a method for real-time animation of large groups of legged creatures. Their approach required the user to manually design the simplified models for each level of detail. Subsequent work has sought to generate approximated models automatically, for example for particle systems [18], plants [19], and hair [20].

Alternatively, one can modify the simulation resolution based on the viewpoint without changing the underlying model. Barran [21] performed fluid simulation on a non-Cartesian grid based on cylindrical coordinates centered at the viewer, thus directly taking the distance from the viewer into account in the discretization. Viewpoint information has also been used to vary simulation resolution in traditional adaptive discretizations for fluids, such as octrees [22], [23] and adaptive particles [24].

In our work, we take inspiration from geometric level of detail techniques, such as those for real-time rendering of terrain [25], [26] or complex scenes like architectural walk-throughs [27]. These techniques inform our understanding of the important view-dependent criteria for representing geometrical detail. Hoppe [28] used surface orientation and screen-space geometric error as refinement criteria. Xia et al. [29] further propose the use of local illumination gradients, projected lengths of edges in screen space, and silhouette boundaries.

3 Methods

Our view-dependent adaptive remeshing scheme builds on ARCSim, the adaptive anisotropic remeshing framework described by Narain et al. [14]. We introduce a new view-dependent refinement strategy that complements their use of dynamical and geometrical refinement criteria. This approach realizes significant speed improvements by extending the domain of adaptive remeshing to include *perceptual* properties as well as physical ones.

The method of Narain et al. defines a *sizing field* that specifies the desired spatially varying resolution of the simulation mesh, taking into account various geometric and dynamic criteria such as local curvature, velocity gradient, compressive strain, and obstacle proximity. It is represented as a 2×2 symmetric tensor field $\mathbf{M}$ which is first computed on faces and then transferred to vertices via area-weighted averaging. Once the sizing field is defined, an edge between vertices i and j is considered valid if its size with respect to $\mathbf{M}$,

$$s(i,j)^2 = \mathbf{u}_{ij}^\mathsf{T} \left(\frac{\mathbf{M}_i + \mathbf{M}_j}{2} \right) \mathbf{u}_{ij} \ , \tag{1}$$

does not exceed 1, where $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ is the vector between the two vertices in material space. If $s(i,j)^2 > 1$, the edge is deemed invalid and must be split. The remeshing algorithm proceeds by splitting all invalid edges, collapsing as many edges as possible without introducing new invalid edges, and flipping edges to maintain an anisotropically Delaunay triangulation. This procedure produces a mesh that is as coarse as possible while containing no invalid edges and remaining Delaunay in the non-Euclidean space of the metric.

We modify their algorithm so that, rather than using purely physical and geometrical criteria to determine the mesh resolution, we vary the desired degree of refinement over space and time based on visual importance relative to a specified camera motion. We implement this variation by modifying the sizing field $\mathbf{M}$ so that the size of each face is no more than what is needed to resolve visually important features. This modification reduces computational effort in regions that are less visible from the camera, bringing about a more efficient simulation without losing significant visual detail.

Our implementation considers two visibility criteria. In regions that are not visible from the current camera position, that is, those that are out of frame or facing away from the camera, we scale the sizing field to uniformly coarsen the mesh. In regions that are visible, we control the sizes of elements in terms of their projected lengths in screen space so that distant or foreshortened elements are coarser. This approach is roughly equivalent to adaptivity based on screen-space metrics. However, to avoid artifacts that would occur due to fast-moving cameras or cuts between views, our algorithm applies conservative spatial and temporal smoothing to the sizing field.

3.1 Coarsening of non-visible regions

For non-visible regions, we seek to uniformly coarsen the mesh relative to the original view-independent sizing field. To do so, we define a scalar field $\nu \leq 1$, which we call the *view factor*, and modify the sizing field as

$$\mathbf{M}_{\text{vd}} = \nu^2 \mathbf{M}. \tag{2}$$

As the sizing criterion (1) is quadratic, this scaling increases the target edge lengths by a factor of ν^{-1}.

In general, we would like to use full resolution ($\nu = 1$) for faces that are visible in the current view, and coarser resolution for back-facing and out-of-frame faces based on

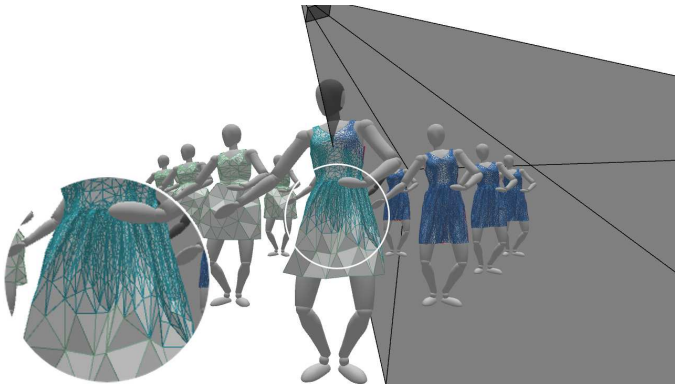

Fig. 2. An example of spatial smoothing. Cyan edges indicate out-of-view faces under spatial smoothing, and green edges indicate out-of-view faces far from the view frustum.

user-specified parameters $\nu_{\text{back}}, \nu_{\text{out}} < 1$. However, simply defining ν in a piecewise-constant fashion causes severe artifacts because of the discontinuous change in the sizing field. First, the discontinuity in sizes at the boundary between in-frame and out-of-frame faces leads to noticeable motion artifacts such as popping due to the influence of spurious forces from the out-of-frame region. Second, rapid camera movements and jump cuts can cause previously coarse regions with inaccurate geometry to suddenly become visible. To eliminate these discontinuities and obtain artifact-free results, we define the view factor in a way that is continuous over both space and time.

3.1.1 Spatial smoothing

Instead of using a spatially discontinuous field, we enforce a continuous falloff of the view factor between in-frame and out-of-frame faces. For a given mesh face, let d be its distance from the view frustum in world space. We define the spatially smoothed view factor $\tilde{\nu}$ by linearly interpolating to ν_{out} over a user-specified margin length m:

$$\tilde{\nu} = \begin{cases} \nu_{\text{fb}} & \text{if } d = 0, \\ \nu_{\text{fb}} - \frac{d}{m}(\nu_{\text{fb}} - \nu_{\text{out}}) & \text{if } 0 < d < m, \\ \nu_{\text{out}} & \text{if } d \geq m, \end{cases} \tag{3}$$

where ν_{fb} is 1 or ν_{back} depending on the direction of the face normal. Thus, we have $\tilde{\nu} = 1$ or ν_{back} for in-frame faces and $\tilde{\nu} = \nu_{\text{out}}$ for out-of-frame faces far from the view frustum, with a continuous transition region in between, as Figure 2 shows.

There is still a discontinuity on the boundary between front-facing faces and backward-facing faces. While it is tempting to use the direction of the normal of back-faces to create a smooth transition, we find that normals can vary too rapidly across silhouettes to offer any useful smoothing that way. Instead, we address this discontinuity with a different approach, described later in Section 3.3.

3.1.2 Temporal smoothing and anticipation

We use temporal smoothing to avoid visibly discontinuous changes in mesh size due to camera motion, which may cause noticeable popping artifacts. We include anticipation that ensures the cloth gets refined to sufficient resolution *before* it appears in the frame, preventing undesirable transients.

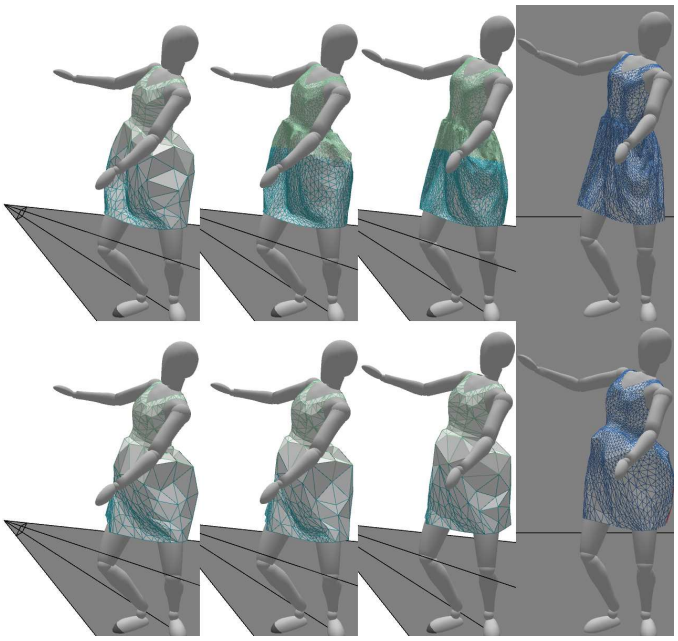

Fig. 3. A sequence of frames with a jump cut, showing a dress simulated with temporal anticipation (above), and without (below). Blue edges indicate faces visible in the current view. Temporal anticipation allows the cloth to reach the correct high-resolution state before it becomes visible.

For any given face, we can treat the view factor before temporal smoothing $\tilde{\nu}$ as a function of time, holding the face fixed and considering the prescribed motion of the viewpoint. We smooth $\tilde{\nu}$ over a time interval $[t, t+T]$ based on the current time t as follows. Define a temporal window function $w(\tau)$ which satisfies $w(0) = 1$ and falls off to zero at $\tau = T$. The temporally smoothed view factor is

$$\nu(t) = \max_{\tau \in [0,T]} w(\tau)\tilde{\nu}(t+\tau). \tag{4}$$

This is analogous to dilation by a non-flat structuring element in mathematical morphology. In our implementation, we use $w(\tau) = 1 - \tau/T$.

Unlike smoothing by, say, moving averages, our approach is conservative in that $\nu(t) \geq \tilde{\nu}(t)$; in particular, visible regions always have $\nu = 1$. Further, in the presence of discontinuities such as jump cuts, ν increases continuously from ν_{out} to 1 over a time period T in advance of the jump. This anticipatory refinement allows enough time for the system to settle into a feasible high-resolution state before becoming visible, as shown in Figure 3.

3.2 Screen-space resolution of visible regions

In non-visible regions, it is sufficient to uniformly coarsen the mesh resolution as above. However, for visible regions, we wish to preserve the geometrical and dynamical detail resolved by the original sizing field as much as possible, only coarsening when such detail would not be visually important.

The sizing field $\mathbf{M}_{\text{vd}}$ gives the mesh resolution needed to accurately capture the dynamics of all cloth features. From the camera's perspective, however, wrinkles and folds that are very distant or foreshorted will appear extremely small in the final image. Such features are not likely to affect the

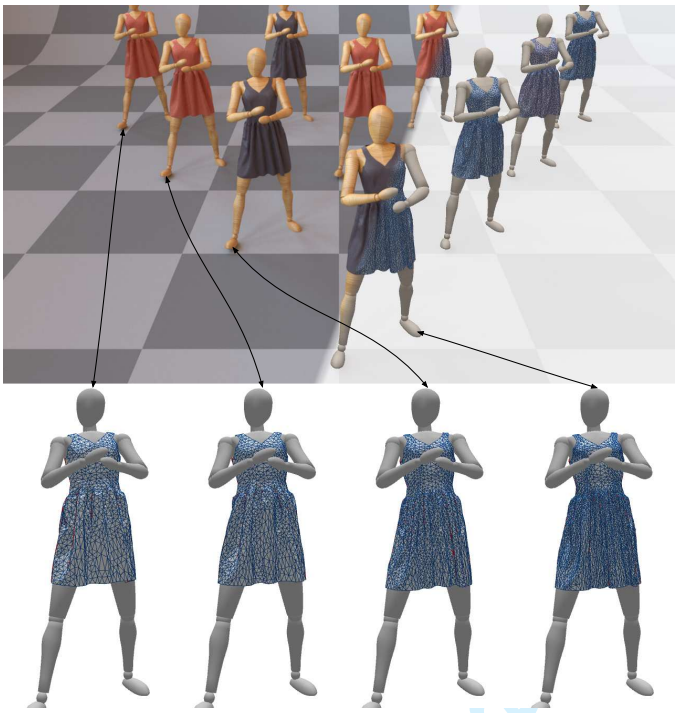

Fig. 4. As the distance between the camera and each character increases from right to left, the simulation mesh becomes progressively coarser (below). However, the minimum edge length measured in screen-space remains roughly constant. As a result the figures shown in the rendered image (above) have a uniform level of visual detail.

visual appearance of the cloth and need not be represented accurately; they can simply be coarsened away. On the other hand, features that appear large in the image should remain accurately represented at the resolution determined by the original sizing field.

In the previous work of Narain et al. [14], the allowed edge lengths were constrained to a range $[\ell_{\min}, \ell_{\max}]$ in material space by clamping the eigenvalues of the sizing tensor to fall between $\ell_{\max}^{-2}$ and $\ell_{\min}^{-2}$. As a result, features that are extremely small in absolute geometrical size are coarsened, while larger ones remain unaffected. In our work, we take the same approach, but apply it to the projected lengths of edges in screen space: edges that *appear* too small, relative to the viewer, are coarsened. To do so, we transform the sizing tensor $\mathbf{M}_{\text{vd}}$ to screen space, apply the bound on the shortest allowable edge length, and transform it back to material space.

For a given configuration of the sheet, consider the function from material-space coordinates of vertices to their screen-space coordinates. On visible faces, this function is locally invertible and its Jacobian $\mathbf{S}$ is a full-rank 2×2 matrix that can be evaluated locally for each face. As the sizing tensor $\mathbf{M}_{\text{vd}}$ acts as a quadratic form acting on vectors in material space, the corresponding tensor that acts on screen-space vectors can be obtained via the transformation

$$\mathcal{S}(\mathbf{M}_{\text{vd}}) = \mathbf{S}^{-\mathsf{T}} \mathbf{M}_{\text{vd}} \mathbf{S}^{-1}. \tag{5}$$

We require the minimum change to the screen-space sizing tensor $\tilde{\mathbf{M}}_{\text{vd}} = \mathcal{S}(\mathbf{M}_{\text{vd}})$ such that edges of screen-space length $\ell_{\min}$ will not be refined further. This modification

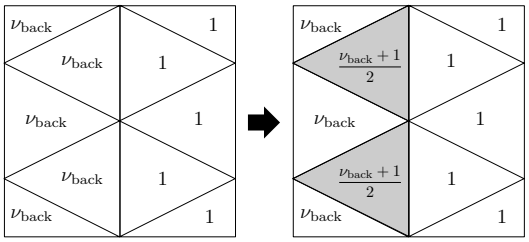

Fig. 5. Locally interpolating discontinuous view factors at silhouettes to ensure smooth silhouettes.

can be achieved by clamping the eigenvalues of $\tilde{\mathbf{M}}_{\text{vd}}$ to not exceed $\ell_{\min}^{-2}$:

$$\tilde{\mathbf{M}}_{\text{vd}} = \mathbf{Q} \begin{bmatrix} \tilde{\lambda}_1 & 0 \\ 0 & \tilde{\lambda}_2 \end{bmatrix} \mathbf{Q}^{\mathsf{T}}, \tag{6}$$

$$\hat{\lambda}_i = \min(\tilde{\lambda}_i, \ell_{\min}^{-2}), \tag{7}$$

$$\hat{\mathbf{M}}_{\text{vd}} = \mathbf{Q} \begin{bmatrix} \hat{\lambda}_1 & 0 \\ 0 & \hat{\lambda}_2 \end{bmatrix} \mathbf{Q}^{\mathsf{T}}. \tag{8}$$

We transform this modified screen-space tensor back into material space to obtain the final sizing field we use for remeshing visible regions, $\mathcal{S}^{-1}(\hat{\mathbf{M}}_{\text{vd}}) = \mathbf{S}^{\mathsf{T}} \hat{\mathbf{M}}_{\text{vd}} \mathbf{S}$.

3.3 Transferring sizing field from faces to vertices

The sizing field defined by the procedures above is represented as a metric tensor on each face. This tensor field must be resampled onto mesh vertices so that it can be used in the sizing criterion (1). Previous work [14] has used a simple area-weighted averaging procedure. However, we have found that that approach tends to lose detail in regions with rapid variation in the sizing field, such as at silhouettes where the view factor changes from 1 to ν_{out}. The issue is exacerbated because coarser regions, which have larger faces, are given higher weight, leading to excessive coarsening at boundary vertices.

In order to handle this discontinuity in view factors, we first resample per-face view factors before the simple area-weighted averaging procedure. If the values of ν differ by more than a specified threshold across any edge, we do simple averaging between two adjacent view factors, and assign the averaged value to one of the faces as shown in Figure 5. As we don't want to change the view factors of the visible faces from 1, we always assign the averaged view factor to a non-visible face.

This approach ensures that silhouette boundaries are refined to the same extent as other visible regions of the cloth, improving the appearance of silhouettes. This change affects the simulation mesh only in a limited area near silhouette boundaries, so it does not hinder overall performance.

3.4 Interpenetration handling

Remeshing invariably produces changes in the geometry of the cloth mesh, and can introduce interpenetrations of the cloth with itself or with obstacles. We found that the simple approach for interpenetration handling used in previous remeshing work [14] does not always converge to an interpenetration-free configuration in the presence

of the aggressive coarsening we perform in non-visible regions. Instead we use the approach of intersection contour minimization proposed by Volino et al. [30], which we found to be robust to large and complex interpenetrations.

3.5 Coarse-scale buckling compensation

Real cloth materials typically have a relatively high resistance to in-plane compression, but a relatively low resistance to out-of-plane bending. As a result, cloth will often drape in configurations that would be non-developable at coarse scale, but where fine-scale wrinkles and folds take the place of coarse-scale compression. These fine wrinkles and folds not only allow commonly observed draping, but they are also responsible for giving specific materials their distinctive appearances.

However, extremely coarse mesh regions do not have the resolution required to form fine wrinkles or folds, and furthermore those coarse regions tend to suffer from element locking that suppresses bending and wrinkle formation. As a result, when out-of-view regions are heavily coarsened, they often exhibit behavior that is dramatically different from that of more refined regions. In particular, coarse regions have a tendency to balloon out in an undesirable manner that is atypical of real cloth, as shown in Figures 3 and 10.

Because extreme coarsening mainly occurs for out-of-frame cloth regions, this undesirable behavior is generally not observable by the viewer. Additionally, our temporal smoothing causes cloth regions to be refined before they are visible, allowing transient artifacts to settle out. Nevertheless, the incorrect motion of out-of-frame regions may cause undesirable side effects, such as producing incorrect dynamics in the stiffened regions to affect the behavior of the visible part of the cloth.

We can avoid this artificial stiffening by switching coarse faces to a material model that permits compression in lieu of sub-element buckling [31], [32]. However, simply removing compression resistance for all out-of-view faces would also create undesirable artifacts, as shown in Figure 6 (middle). Instead, we allow compression adaptively based on each face's area as follows.

The Green strain tensor of face i is

$$\mathbf{E}_i = \frac{1}{2}(\mathbf{F}_i^{\mathsf{T}}\mathbf{F}_i - \mathbf{I}), \tag{9}$$

where $\mathbf{F}_i$ is the deformation gradient. We reduce the material's compression resistance by simply reducing the amount of compressive strain that the constitutive model sees. Specifically, for each out-of-frustum face, we compute the eigendecomposition of its strain tensor $\mathbf{E}_i$ to obtain principal strains λ_1 and λ_2. We replace each principal strain with

$$\hat{\lambda}_j = \begin{cases} \lambda_j & \text{if } \lambda_j \geq 0, \\ \beta_i \lambda_j & \text{if } \lambda_j < 0, \end{cases} \tag{10}$$

with area-based compression factor β_i,

$$\beta_i = \left(\frac{b - A_i}{b - a}\right)^k, \tag{11}$$

$$a = \min_{\forall j \in S} A_j, \quad b = \max_{\forall j \in S} A_j, \tag{12}$$

where A_i is area of face i in material space, S is a set of all out-of-frustum faces, and $k \geq 1$ is a parameter controlling the amount of compensation. Because this compression factor increases with material-space area, the degree of allowed compression changes adaptively: the bigger the triangle, the more compressible it is. This behavior approximates the fine-scale buckling that otherwise have occurred in the cloth represented by the coarse element.

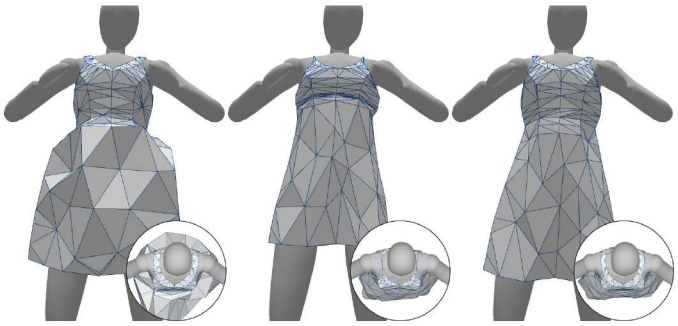

Fig. 6. A comparison of no compensation (left), non-adaptive compensation (middle), and adaptive compensation with $k = 10$ (right).

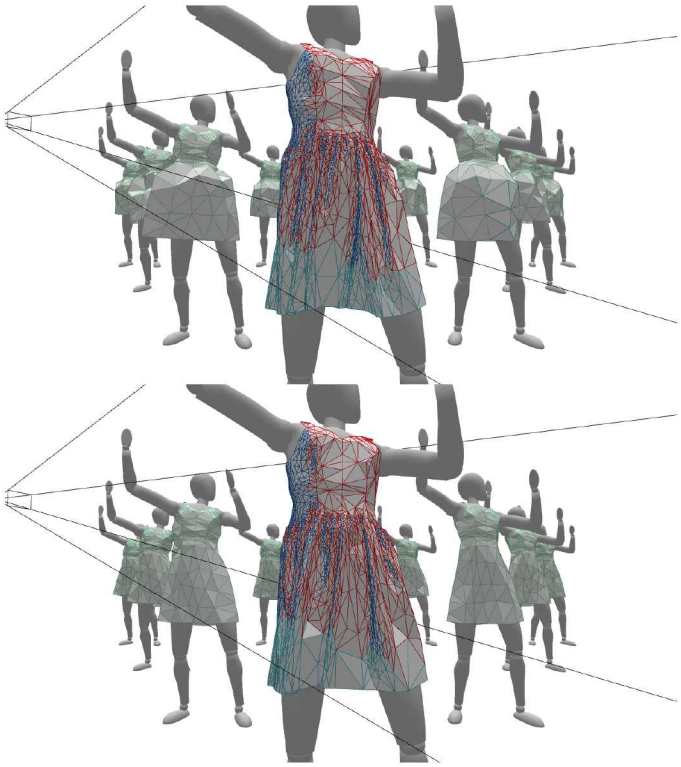

Fig. 7. No compensation (top) and adaptive compensation with $k = 10$ (bottom) in Multiple Dancers example.

We note that this approach to approximating the coarse effects of fine-scale buckling is not specific to view-dependent adaptivity, and it could be applied to other adaptive algorithms suffering from similar artifacts. The compression factor β is unrelated to the view factor, as buckling is purely a physical phenomenon independent of the viewing direction. Therefore, we have defined the compression factor entirely in terms of the area of faces and a user-defined parameter k.

TABLE 1
Statistics and timing numbers for the examples. Non-adaptive simulations use a fixed high-resolution mesh. Adaptive simulations use the unmodified scheme implemented in ARCSim. View-dependent simulations use the method described in this paper. The adaptive simulations are used as a baseline for comparison. The non-adaptive mesh resolution was selected to match the visual quality of the adaptive simulations.

	Num. Faces			Num. Vertices			Time / Frame (seconds)			Speed-up
	min	max	mean	min	max	mean	min	max	mean	
Karate										
Non-adaptive	123,790	123,790	123,790	63,889	63,889	63,889	136.47	288.01	161.44	0.49×
Adaptive	25,785	71,142	41,973	14,135	37,199	22,358	31.41	184.39	79.04	1×
View-dependent	5,223	39,139	21,501	3,142	20,770	11,670	5.52	83.29	33.02	2.39×
Solo Dancer										
Non-adaptive	43,710	43,710	43,710	22,736	22,736	22,736	27.31	58.09	29.27	0.54×
Adaptive	12,030	21,763	18,041	6,593	11,535	9,659	7.78	22.00	15.80	1×
View-dependent	730	15,951	9,638	560	8,599	5,314	0.83	13.81	7.55	2.09×
Multiple Dancers										
Non-adaptive	437,100	437,100	437,100	227,360	227,360	227,360	273.13	580.85	292.73	0.47×
Adaptive	119,515	184,340	161,028	65,505	98,630	86,897	79.97	178.24	136.99	1×
View-dependent	11,216	102,945	36,339	7,619	56,285	21,228	12.02	82.54	30.76	4.45×

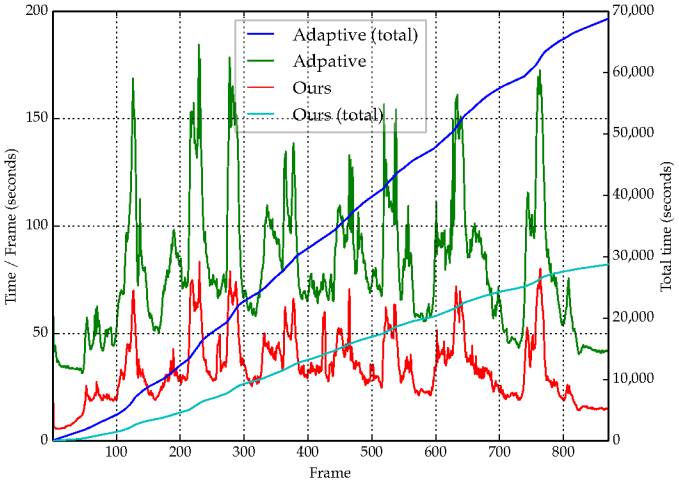

Fig. 8. A comparison of view-independent and view-dependent adaptive simulations for the example of Figure 1 in terms of per-frame and cumulative simulation times.

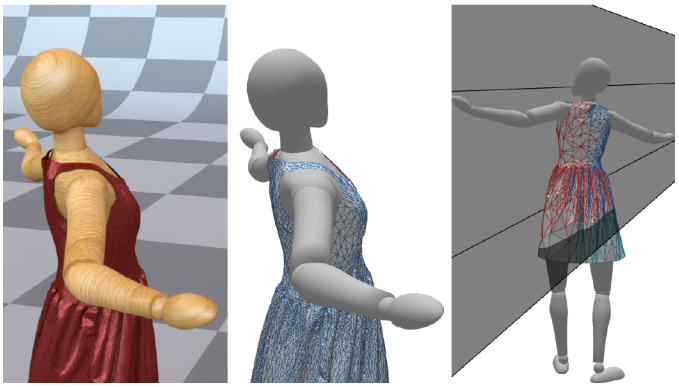

Fig. 9. A solo dancer is shown from the camera's perspective (left, middle) and from an external perspective with the camera's view frustum drawn in black wireframe (right).

4 Results and Discussion

We have implemented the methods described in this paper as an extension to the ARCSim simulation code. These extensions will be integrated into the v0.3.0 distribution.

To test the utility of the proposed methods, we ran comparisons with adaptive and non-adaptive simulations on several examples. All simulations were run on a machine with an Intel Xeon E5-2680 v2 processor and 16 GB RAM using 4 execution threads. The constants for view-dependent refinement that were used for these examples are $\nu_{\text{back}} = 0.2$, $m = 0.4$ m, $T = 5$ frames, and $\nu_{\text{out}} = 0.01$. However, for the example shown in Figure 1, we used larger $\nu_{\text{out}} = 0.1$ to avoid intersection problems with the layered garments.

Figure 1 shows a character wearing a layered outfit consisting of three interacting garments. As the accompanying video shows, the character's clothing exhibits complex dynamic motion with detailed wrinkles. As the camera moves continuously or transits between cuts, the simulation mesh is updated to maintain a constant level of visible detail while out-of-view regions are aggressively coarsened. Figure 8 plots the time per frame and total cumulative computation time for this example for both the basic adaptive simulation and our view-dependent version.

Figure 9 shows a character wearing a simple dress, showing a degree of view-dependent refinement and coarsening similar to Figure 1. A group of ten such characters is shown in Figure 10. Note that while these characters are all performing the same actions in the same type of dress, they are simulated individually with different resolutions depending on their location relative to the viewpoint. In practical applications, multiple characters in a crowd would likely have different garments and distinct movements, and would therefore also have to be simulated individually even without view dependence.

Timings for these three examples are reported in Table 1. The single character examples realize a speed-up between 2.1× and 2.4×. This speed-up becomes more substantial for the group of ten characters where it is roughly 4.5×. The greater speed-up occurs because when a single character fills the screen, requiring full adaptive resolution, it tends to obscure others which then switch to low resolution and generate substantial savings. In order for all characters to

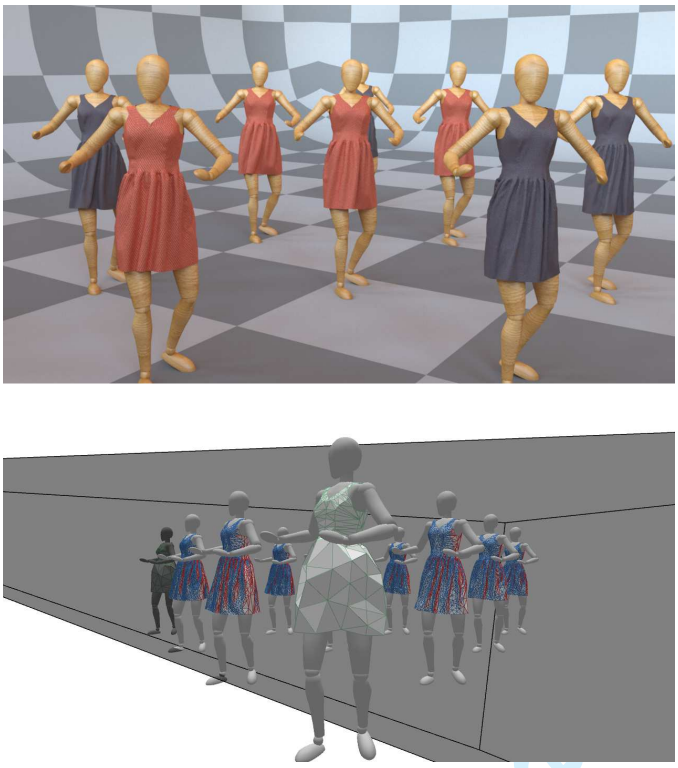

Fig. 10. A group of multiple dancers is rendered from the camera's perspective (top), while an external view showing the camera's view frustum is shown below. Characters completely outside the view frustum are simulated at very low resolution.

be visible, the camera must be fairly far back which yields savings across all characters. These same factors would hold more strongly for larger groups. Accordingly we hypothesize that massive scenes involving thousands of characters would likely realize speed-ups substantially larger than those we have observed for small groups.

Remeshing does introduce small amounts of motion that might be visible if the character is otherwise still. As shown in the video, however, this motion does not manifest as visible popping because the dramatic remeshing generally happens off screen. The motion that appears on screen is only gentle swaying that looks fairly natural. If the subject is moving, then this gentle motion is completely masked by the normal motion of the cloth. Even with a still subject, the artificial motion can be hard to detect due to the movement of the camera.

5 Conclusions

The methods we have described provide a simple way of achieving computational savings for cloth simulation. For scenes involving multiple characters or large crowds, these savings can be substantial. We have demonstrated our approach in simulations of clothing, but believe that it could equally well be applied to other objects that can be simulated in an adaptive framework, including materials that can crumple [15] or fracture [16].

The main limitation of our method is that it requires an adaptive framework. However, once that framework is in place, view-dependent adaptivity is relatively simple to implement. For the group of dancers, our view-dependent adaptive simulation is nearly $10\times$ faster than an equivalent non-adaptive simulation. We believe that such large performance gains outweigh any costs associated with changing mesh topology. We also believe that our approach would scale very well to massive scenes with thousands of actors, where it would produce even larger savings.

As mentioned in the introduction, our method targets high-quality offline simulation, and there would be additional challenges in applying it to interactive animation. In particular, our temporal smoothing requires a prescribed camera motion, so it may not be possible to apply it in interactive settings where camera motions are not predetermined.

In general, simulations used for physically based animation in graphics have been designed so that they capture visible phenomena for realistic appearance. These simulations typically avoid the type of error analysis that one finds in most engineering contexts because it is difficult to quantify a measure of perceptual error that would be relevant to graphics simulations. The work we've presented here explicitly drives adaption according to a heuristic measure of visible error. Future work in this area could more explicitly take perceptual error into account so that reduced resolution could be used where masking effects due to motion, complexity, or other phenomena cause humans to be less sensitive to apparent detail.

Acknowledgments

We thank the other members of the Visual Computing Lab at UC Berkeley for their valuable advice and help. This work was supported by the Intel Science and Technology Center for Visual Computing (ISTC-VC), an AWS in Education Grant award, and a gift from Pixar Animation Studios. Woojong Koh was supported by a Samsung Scholarship.

We built our method on top of the ARCSim framework developed in UC Berkeley, and the images in this paper and video were rendered with the Mitsuba renderer by Wenzel Jakob.

References

[1] F. Losasso, F. Gibou, and R. Fedkiw, "Simulating water and smoke with an octree data structure," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 457–462, Aug. 2004.

[2] B. Adams, M. Pauly, R. Keiser, and L. J. Guibas, "Adaptively sampled particle fluids," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[3] N. Chentanez and M. Müller, "Real-time Eulerian water simulation using a restricted tall cell grid," in *ACM SIGGRAPH 2011 Papers*, ser. SIGGRAPH 2011, 2011, pp. 82:1–82:10.

[4] B. M. Klingner, B. E. Feldman, N. Chentanez, and J. F. O'Brien, "Fluid animation with dynamic meshes," in *Proceedings of ACM SIGGRAPH 2006*, Aug. 2006, pp. 820–825.

[5] N. Chentanez, B. E. Feldman, F. Labelle, J. F. O'Brien, and J. R. Shewchuk, "Liquid simulation on lattice-based tetrahedral meshes," in *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2007*, Aug. 2007, pp. 219–228.

[6] R. Ando, N. Thürey, and C. Wojtan, "Highly adaptive liquid simulations on tetrahedral meshes," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 103:1–103:10, Jul. 2013.

[7] D. Hutchinson, M. Preston, and T. Hewitt, "Adaptive refinement for mass/spring simulations," in *7th Eurographics Workshop on Animation and Simulation*. Springer-Verlag, 1996, pp. 31–45.

[8] S. Redon, N. Galoppo, and M. C. Lin, "Adaptive dynamics of articulated bodies," *ACM Transactions on Graphics (SIGGRAPH 2005)*, vol. 24, no. 3, 2005.
[9] E. Grinspun, P. Krysl, and P. Schröder, "CHARMS: A simple framework for adaptive simulation," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH 2002, 2002, pp. 281–290.
[10] J. A. Thingvold and E. Cohen, "Physical modeling with B-spline surfaces for interactive design and animation," *SIGGRAPH Comput. Graph.*, vol. 24, no. 2, pp. 129–137, Feb. 1990.
[11] L. Li and V. Volkov, "Cloth animation with adaptively refined meshes," in *Proc. 28th Australasian Computer Science Conference*, vol. 38, 2005.
[12] J. Villard and H. Borouchaki, "Adaptive meshing for cloth animation," *Engineering with Computers*, vol. 20, no. 4, pp. 333–341, 2005.
[13] T. Brochu, E. Edwards, and R. Bridson, "Efficient geometrically exact continuous collision detection," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 96:1–96:7, Jul. 2012.
[14] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive anisotropic remeshing for cloth simulation," *ACM Transactions on Graphics*, vol. 31, no. 6, pp. 147:1–10, Nov. 2012, proceedings of ACM SIGGRAPH Asia 2012, Singapore.
[15] R. Narain, T. Pfaff, and J. F. O'Brien, "Folding and crumpling adaptive sheets," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 51:1–51:8, Jul. 2013.
[16] T. Pfaff, R. Narain, J. M. de Joya, and J. F. O'Brien, "Adaptive tearing and cracking of thin sheets," *ACM Trans. Graph.*, 2014, to appear.
[17] D. A. Carlson and J. K. Hodgins, "Simulation levels of detail for real-time animation," in *Proceedings of the Conference on Graphics Interface '97*, 1997, pp. 1–8.
[18] D. O'Brien, S. Fisher, and M. Lin, "Automatic simplification of particle system dynamics," in *The Fourteenth Conference on Computer Animation. Proceedings*, 2001, pp. 210–257.
[19] J. Beaudoin and J. Keyser, "Simulation levels of detail for plant motion," in *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '04, 2004, pp. 297–304.
[20] K. Ward, M. C. Lin, J. Lee, S. Fisher, and D. Macri, "Modeling hair using level-of-detail representations," in *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, ser. CASA '03, 2003, pp. 41–47.
[21] B. A. Barran, "View dependent fluid dynamics," Master's thesis, Texas A&M University, 2006.
[22] J. Kim, I. Ihm, and D. Cha, "View-dependent adaptive animation of liquids," *ETRI Journal*, vol. 28, no. 6, pp. 697–708, Dec. 2006.
[23] R. Bunlutangtum and P. Kanongchaiyos, "Enhanced view-dependent adaptive grid refinement for animating fluids," in *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, ser. VRCAI '11, 2011, pp. 415–418.
[24] B. Solenthaler and M. Gross, "Two-scale particle simulation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 81:1–81:8, Jul. 2011.
[25] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "ROAMing terrain: Real-time optimally adapting meshes," in *Proceedings of the 8th Conference on Visualization '97*, ser. VIS '97, 1997, pp. 81–88.
[26] H. Hoppe, "Smooth view-dependent level-of-detail control and its application to terrain rendering," in *Proceedings of the Conference on Visualization '98*, ser. VIS '98, 1998, pp. 35–42.
[27] T. A. Funkhouser and C. H. Séquin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '93, 1993, pp. 247–254.
[28] H. Hoppe, "View-dependent refinement of progressive meshes," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97, 1997, pp. 189–198.
[29] J. C. Xia, J. El-Sana, and A. Varshney, "Adaptive real-time level-of-detail-based rendering for polygonal models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 171–183, Apr. 1997.
[30] P. Volino and N. Magnenat-Thalmann, "Resolving surface collisions through intersection contour minimization," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1154–1159, Jul. 2006.
[31] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 604–611, Jul. 2002.
[32] H. Wang, J. O'Brien, and R. Ramamoorthi, "Multi-resolution isotropic strain limiting," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 156:1–156:10, Dec. 2010.

PLACE PHOTO HERE

Woojong Koh is a third-year Ph.D. student in Computer Science at the University of California, Berkeley, and received his B.S. in Computer Science and Engineering at the Seoul National University, Seoul, South Korea.

PLACE PHOTO HERE

Rahul Narain is a postdoctoral scholar at the University of California, Berkeley, and received his Ph.D. in computer science from the University of North Carolina at Chapel Hill. His research interests lie in physically-based animation, particularly focusing on numerical techniques for modeling complex multiscale phenomena. Rahul Narain will join the University of Minnesota, Twin Cities as an assistant professor in 2015.

PLACE PHOTO HERE

James F. O'Brien is a Professor of Computer Science at the University of California, Berkeley. His primary area of interest is Computer Animation, with an emphasis on generating realistic motion using physically based simulation and motion capture techniques. He has authored numerous papers on these topics. In addition to his research pursuits, Prof. O'Brien has worked with several game companies on integrating advanced simulation physics into game engines, and his methods for destruction modeling have been used in more than 58 feature films. He received his doctorate from the Georgia Institute of Technology in 2000, the same year he joined the Faculty at U.C. Berkeley. Professor O'Brien is a Sloan Fellow and ACM Distinguished Scientist, Technology Review selected him as one of their TR-100, and he has been awarded research grants from the Okawa and Hellman Foundations. He is currently serving as ACM SIGGRAPH Director at Large.

Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014)
Vladlen Koltun and Eftychios Sifakis (Editors)

View-Dependent Adaptive Cloth Simulation

Woojong Koh Rahul Narain James F. O'Brien

University of California, Berkeley

Abstract

This paper describes a method for view-dependent cloth simulation using dynamically adaptive mesh refinement and coarsening. Given a prescribed camera motion, the method adjusts the criteria controlling refinement to account for visibility and apparent size in the camera's view. Objectionable dynamic artifacts are avoided by anticipative refinement and smoothed coarsening. This approach preserves the appearance of detailed cloth throughout the animation while avoiding the wasted effort of simulating details that would not be discernible to the viewer. The computational savings realized by this method increase as scene complexity grows, producing a $2\times$ *speed-up for a single character and more than* $4\times$ *for a small group.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.

1. Introduction

Cloth simulation for visual effects has reached a mature state where the use of virtual characters wearing simulated clothing is now widespread. However, cloth simulation remains computationally expensive, particularly when films require high-quality, realistic results computed at high resolution. For characters that are far from the camera, or otherwise less visible in a shot, fine details will not be visible to the viewer and work spent computing those details is wasted. In most film production settings, both the camera and character motions are known before the simulations are run, and one could use this information to substitute cheaper low-resolution simulations on distant or out-of-frame characters. Unfortunately, manually swapping some characters to low-resolution simulations is cumbersome, particularly when a single character's clothing requires high resolution for some parts of a shot but not others, but the cloth motion must appear coherent throughout. For closeup shots where only part of a character is in frame, savings could also be realized by reducing the computation devoted to out-of-frame cloth, so long as such savings do not result in dynamic artifacts that affect visible parts of the cloth.

In this paper we describe a method for *view-dependent simulation* using dynamically adaptive mesh refinement and coarsening. Instead of using a fixed-resolution simulation mesh, the mesh undergoes local adaptation based on the geometric and dynamic detail of the simulated cloth. The degree to which this detail is resolved is adjusted locally based on the view of the simulated cloth. Areas that appear large in the camera will be refined to show finely detailed dynamics. Areas that are out of frame, facing away from the camera, or at a distance will be correspondingly coarser.

Figure 1: *The clothing on this character uses view-dependent simulation. Visible areas (blue) are simulated at high resolution, as measured in screen-space. Areas not visible to the camera (red) are simulated at reduced resolution. The first two images show views from the camera's perspective. The rightmost image shows an outside perspective with the camera's view frustum drawn in black wireframe.*

The goal of this work is to preserve the appearance of detailed simulation throughout the animation while avoid-

ing the wasted effort of simulating details that will not be apparent to the viewer. Further, there should be no visible dynamic artifacts created due to varying refinement as the camera and objects move about. Finally, cloth that leaves and reenters visibility should appear to have coherent and consistent dynamics.

Our work builds on the publicly available ARCSim framework (http://graphics.berkeley.edu/resources/ARCSim) which can be used to animate sheets of deformable materials such as cloth, paper, plastic, and metal. ARCSim adaptively refines anisotropic triangle meshes to efficiently resolve the geometric and dynamic detail of the simulated objects. Our method modifies the metrics used by ARCSim so that local mesh visibility is accounted for during refinement. Given the existing framework for adaptivity, our view-dependent refinement is easy to implement and has negligible overhead.

In cases where only a single character is being modeled, we realize modest savings of roughly a $2.4\times$ speed-up in comparison with ARCSim's default adaptive simulation. These savings are due to coarsening out-of-view and back-facing parts of the character's clothing. For small crowd scenes, the savings are larger, up to $4.5\times$, as background and out-of-view characters are automatically coarsened. Compared to a non-adaptive simulation, the speed-up is more than $9\times$. For massive crowd scenes with thousands of agents, we expect that even greater savings could be realized with out approach.

2. Related Work

Adaptive discretizations have been found to give significant performance and scalability benefits for a number of computationally intensive simulation tasks. In fluid simulation, detailed liquid surfaces can be animated efficiently by refining the spatial resolution near the surface using octrees [LGF04], adaptively sampled particles [APKG07], tall-cell grids [CM11], or tetrahedral meshes [KFCO06, CFL*07, ATW13]. Adaptive refinement and simplification techniques have also been proposed for mass-spring systems [HPH96], articulated bodies [RGL05] and finite element models [GKS02]. Most relevant to our work are techniques for adaptive cloth simulation [TC90, LV05, VB05, BEB12, NSO12], which use remeshing to resolve detailed wrinkles and folds. The approach of Narain et al. [NSO12] has also been extended to efficiently model plastic deformation and sharp creases [NPO13] as well as complex fracture patterns [PNdJO14]. However, all those techniques are view-independent and sizing is controlled only by geometrical and dynamical properties of the simulated system.

For animation applications, a number of techniques have also been proposed that take into account the current viewpoint and attempt to expend less computational effort in regions that are visually less important. One approach, known as simulation level of detail, is to switch between dynamical models of varying degrees of simplification, depending on the viewpoint. Carlson and Hodgins [CH97] introduced such a method for real-time animation of large groups of legged creatures. Their approach required the user to manually design the simplified models for each level of detail.Subsequent work has sought to generate approximated models automatically, for example for particle systems [OFL01], plants [BK04], and hair [WLL*03].

Alternatively, one can modify the simulation resolution based on the viewpoint without changing the underlying model. Barran [Bar06] performed fluid simulation on a non-Cartesian grid based on cylindrical coordinates centered at the viewer, thus directly taking the distance from the viewer into account in the discretization. Viewpoint information has also been used to vary simulation resolution in traditional adaptive discretizations for fluids, such as octrees [KIC06, BK11] and adaptive particles [SG11].

In our work, we also take inspiration from geometric level of detail techniques, such as those for real-time rendering of terrain [DWS*97, Hop98] or complex scenes like architectural walkthroughs [FS93]. These techniques inform our understanding of the important view-dependent criteria for representing geometrical detail. Hoppe [Hop97] used surface orientation and screen-space geometric error as refinement criteria. Xia et al. [XESV97] further propose the use of local illumination gradients, projected lengths of edges in screen space, and silhouette boundaries.

3. Methods

Our view-dependent adaptive remeshing scheme builds on the adaptive anisotropic remeshing framework described by Narain et al. [NSO12]. We introduce a new view-dependent refinement strategy that complements their use of dynamical and geometrical refinement criteria. This approach realizes significant speed improvements by extending the domain of adaptive remeshing to include *perceptual* properties as well as physical ones.

The method of Narain et al. defines a *sizing field* that specifies the desired spatially varying resolution of the simulation mesh, taking into account various geometric and dynamic criteria such as local curvature, velocity gradient, compressive strain, and obstacle proximity. It is represented as a 2×2 symmetric tensor field $\mathbf{M}$ which is first computed on faces and then transferred to vertices via area-weighted averaging. Once the sizing field is defined, an edge between vertices i and j is considered valid if its size with respect to $\mathbf{M}$,

$$s(i,j)^2 = \mathbf{u}_{ij}^{\mathsf{T}} \left(\frac{\mathbf{M}_i + \mathbf{M}_j}{2} \right) \mathbf{u}_{ij} \,, \tag{1}$$

does not exceed 1, where $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ is the vector between the two vertices in material space. If $s(i,j)^2 > 1$, the edge is deemed invalid and must be split. The remeshing algorithm proceeds by splitting all invalid edges, collapsing as many edges as possible without introducing new invalid edges, and

flipping edges to maintain an anisotropically Delaunay triangulation. This procedure produces a mesh that is as coarse as possible while containing no invalid edges and remaining Delaunay in the non-Euclidean space of the metric.

We modify their algorithm so that, rather than using purely physical and geometrical criteria to determine the mesh resolution, we vary the desired degree of refinement over space and time based on visual importance relative to a specified camera motion. We implement this variation by modifying the sizing field $\mathbf{M}$ so that the size of each face is no more than what is needed to resolve visually important features. This modification reduces computational effort in regions that are less visible from the camera, bringing about a more efficient simulation without losing significant visual detail.

Our implementation considers two visibility criteria. In regions that are not visible from the current camera position, that is, those that are out of frame or facing away from the camera, we scale the sizing field to uniformly coarsen the mesh. In regions that are visible, we control the sizes of elements in terms of their projected lengths in screen space so that distant or foreshortened elements are coarser. This approach is roughly equivalent to adaptivity based on screen-space metrics. However, to avoid artifacts that would occur due to fast-moving cameras or cuts between views, our algorithm applies conservative spatial and temporal smoothing to the sizing field.

3.1. Coarsening of non-visible regions

For non-visible regions, we seek to uniformly coarsen the mesh relative to the original view-independent sizing field. To do so, we define a scalar field $\nu \leq 1$, which we call the *view factor*, and modify the sizing field as

$$\mathbf{M}_{\mathrm{vd}} = \nu^2 \mathbf{M}. \tag{2}$$

As the sizing criterion (1) is quadratic, this scaling increases the target edge lengths by a factor of ν^{-1}.

In general, we would like to use full resolution ($\nu = 1$) for faces that are visible in the current view, and coarser resolution for back-facing and out-of-frame faces based on user-specified parameters $\nu_{\mathrm{back}}, \nu_{\mathrm{out}} < 1$. However, simply defining ν in this piecewise-constant fashion causes severe artifacts because of the discontinuous change in the sizing field. First, the discontinuity in sizes at the boundary between in-frame and out-of-frame faces leads to noticeable motion artifacts such as popping due to the influence of spurious forces from the out-of-frame region. Second, rapid camera movements and jump cuts can cause previously coarse regions with inaccurate geometry to suddenly become visible. To eliminate these discontinuities and obtain artifact-free results, we define the view factor in a way that is continuous over both space and time.

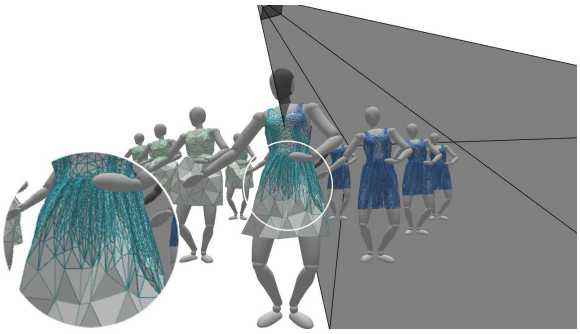

Figure 2: *An example of spatial smoothing. Cyan edges indicate out-of-view faces under spatial smoothing, and green edges indicate out-of-view faces far from the view frustum.*

3.1.1. Spatial smoothing

Instead of using a spatially discontinuous field, we enforce a continuous falloff of the view factor between in-frame and out-of-frame faces. For a given mesh face, let d be its distance from the view frustum in world space. We define the spatially smoothed view factor $\tilde{\nu}$ by linearly interpolating to ν_{out} over a user-specified margin length m:

$$\tilde{\nu} = \begin{cases} \nu_{\mathrm{fb}} & \text{if } d = 0, \\ \nu_{\mathrm{fb}} - \frac{d}{m}(\nu_{\mathrm{fb}} - \nu_{\mathrm{out}}) & \text{if } 0 < d < m, \\ \nu_{\mathrm{out}} & \text{if } d \geq m, \end{cases} \tag{3}$$

where ν_{fb} is 1 or ν_{back} depending on the direction of the face normal. Thus, we have $\tilde{\nu} = 1$ or ν_{back} for in-frame faces and $\tilde{\nu} = \nu_{\mathrm{out}}$ for out-of-frame faces far from the view frustum, with a continuous transition region in between, as Figure 2 shows.

There is still a discontinuity on the boundary between front-facing faces and backward-facing faces. While it is tempting to use the direction of the normal of back-faces to create a smooth transition, we find that normals can vary too rapidly across silhouettes to offer any useful smoothing that way. Instead, we address this discontinuity with a different approach, described later in Section 3.3.

3.1.2. Temporal smoothing and anticipation

We use temporal smoothing to avoid visibly discontinuous changes in mesh size due to camera motion, which may cause noticeable popping artifacts. We include anticipation that ensures the cloth gets refined to sufficient resolution *before* it appears in the frame, preventing undesirable transients.

For any given face, we can treat the view factor before temporal smoothing $\tilde{\nu}$ as a function of time, holding the face fixed and considering the prescribed motion of the viewpoint. We smooth $\tilde{\nu}$ over a time interval $[t, t+T]$ based on the current time t as follows. Define a temporal window function $w(\tau)$ which satisfies $w(0) = 1$ and falls off to zero at $\tau = T$.

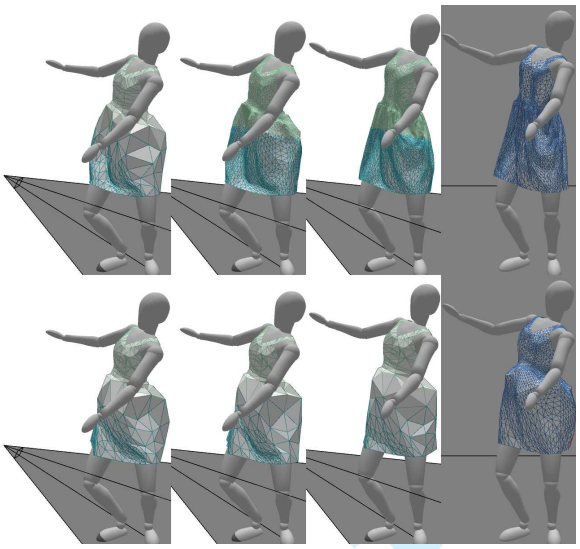

Figure 3: *A sequence of frames with a jump cut, showing a dress simulated with temporal anticipation (above), and without (below). Blue edges indicate faces visible in the current view. Temporal anticipation allows the cloth to reach the correct high-resolution state before it becomes visible.*

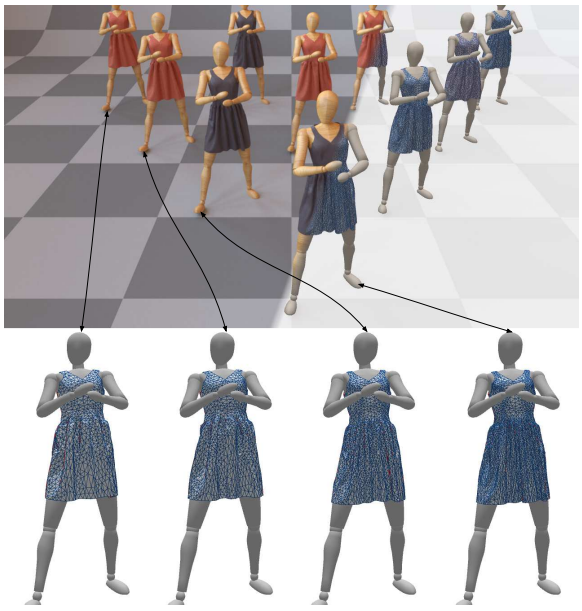

Figure 4: *As the distance between the camera and each character increases from right to left, the simulation mesh becomes progressively coarser (below). However, the minimum edge length measured in screen-space remains roughly constant. As a result the figures shown in the rendered image (above) have a uniform level of visual detail.*

The temporally smoothed view factor is

$$\nu(t) = \max_{\tau \in [0,T]} w(\tau)\tilde{\nu}(t+\tau). \tag{4}$$

This is analogous to dilation by a non-flat structuring element in mathematical morphology. In our implementation, we use $w(\tau) = 1 - \tau/T$.

Unlike smoothing by, say, moving averages, our approach is conservative in that $\nu(t) \geq \tilde{\nu}(t)$; in particular, visible regions always have $\nu = 1$. Further, in the presence of discontinuities such as jump cuts, ν increases continuously from ν_{out} to 1 over a time period T in advance of the jump. This anticipatory refinement allows enough time for the system to settle into a feasible high-resolution state before becoming visible, as shown in Figure 3.

3.2. Screen-space resolution of visible regions

In non-visible regions, it is sufficient to uniformly coarsen the mesh resolution as above. However, for visible regions, we wish to preserve the geometrical and dynamical detail resolved by the original sizing field as much as possible, only coarsening when such detail would not be visually important.

The original sizing field $\mathbf{M}_{\text{vd}}$ gives the mesh resolution needed to accurately capture the dynamics of all cloth features. From the camera's perspective, however, wrinkles and folds that are very distant or foreshorted will appear extremely small in the final image. Such features are not likely to affect the visual appearance of the cloth and need not be represented accurately; they can simply be coarsened away. On the other hand, features that appear large in the image should remain accurately represented at the resolution determined by the original sizing field.

In the previous work of Narain et al. [NSO12], the allowed edge lengths were constrained to a range $[\ell_{\min}, \ell_{\max}]$ in material space by clamping the eigenvalues of the sizing tensor to fall between $\ell_{\max}^{-2}$ and $\ell_{\min}^{-2}$. As a result, features that are extremely small in absolute geometrical size are coarsened, while larger ones remain unaffected. In our work, we take the same approach, but apply it to the projected lengths of edges in screen space: edges that *appear* too small, relative to the viewer, are coarsened. To do so, we transform the sizing tensor $\mathbf{M}_{\text{vd}}$ to screen space, apply the bound on the shortest allowable edge length, and transform it back to material space.

For a given configuration of the sheet, consider the function from material-space coordinates of vertices to their screen-space coordinates. On visible faces, this function is locally invertible and its Jacobian $\mathbf{S}$ is a full-rank 2×2 matrix that can be evaluated locally for each face. As the sizing tensor $\mathbf{M}_{\text{vd}}$ acts as a quadratic form acting on vectors in material space, the corresponding tensor that acts on screen-space vectors can be obtained via the transformation

$$\mathscr{S}(\mathbf{M}_{\text{vd}}) = \mathbf{S}^{-\mathsf{T}}\mathbf{M}_{\text{vd}}\mathbf{S}^{-1}. \tag{5}$$

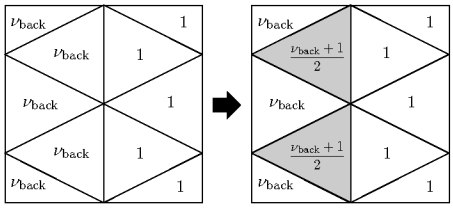

Figure 5: *Locally interpolating discontinuous view factors at silhouettes to ensure smooth silhouettes.*

We require the minimum change to the screen-space sizing tensor $\tilde{\mathbf{M}}_{\text{vd}} = \mathscr{S}(\mathbf{M}_{\text{vd}})$ such that edges of screen-space length ℓ_{min} will not be refined further. This modification can be achieved by clamping the eigenvalues of $\tilde{\mathbf{M}}_{\text{vd}}$ to not exceed ℓ_{min}^{-2}:

$$\tilde{\mathbf{M}}_{\text{vd}} = \mathbf{Q}\begin{bmatrix}\tilde{\lambda}_1 & 0\\ 0 & \tilde{\lambda}_2\end{bmatrix}\mathbf{Q}^{\mathsf{T}}, \tag{6}$$

$$\hat{\lambda}_i = \min(\tilde{\lambda}_i, \ell_{\text{min}}^{-2}), \tag{7}$$

$$\hat{\mathbf{M}}_{\text{vd}} = \mathbf{Q}\begin{bmatrix}\hat{\lambda}_1 & 0\\ 0 & \hat{\lambda}_2\end{bmatrix}\mathbf{Q}^{\mathsf{T}}. \tag{8}$$

We transform this modified screen-space tensor back into material space to obtain the final sizing field we use for remeshing visible regions, $\mathscr{S}^{-1}(\hat{\mathbf{M}}_{\text{vd}}) = \mathbf{S}^{\mathsf{T}}\hat{\mathbf{M}}_{\text{vd}}\mathbf{S}$.

3.3. Transferring sizing field from faces to vertices

The sizing field defined by the procedures above is represented as a metric tensor on each face. This tensor field must be resampled onto mesh vertices so that it can be used in the sizing criterion (1). Previous work [NSO12] has used a simple area-weighted averaging procedure. However, we have found that that approach tends to lose detail in regions with rapid variation in the sizing field, such as at silhouettes where the view factor changes from 1 to ν_{out}. The issue is exacerbated because coarser regions, which have larger faces, are given higher weight, leading to excessive coarsening at boundary vertices.

In order to handle this discontinuity on view factors, we first resample per-face view factors before the simple area-weighted averaging procedure. If the values of ν differ by more than a specified threshold across any edge, we do simple averaging between two adjacent view factors, and assign the averaged value to one of the faces as shown in Figure 5. As we don't want to change the view factors of the visible faces from 1, we always assign the averaged view factor to a non-visible face.

This approach ensures that silhouette boundaries are refined to the same extent as other visible regions of the cloth, improving the appearance of silhouettes. This change affects the simulation mesh only in a limited area near silhouette boundaries, so it does not hinder overall performance.

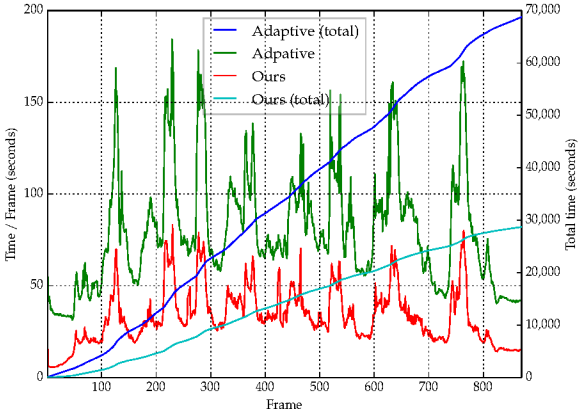

Figure 6: *A comparison of view-independent and view-dependent adaptive simulations for the example of Figure 1 in terms of per-frame and cumulative simulation times.*

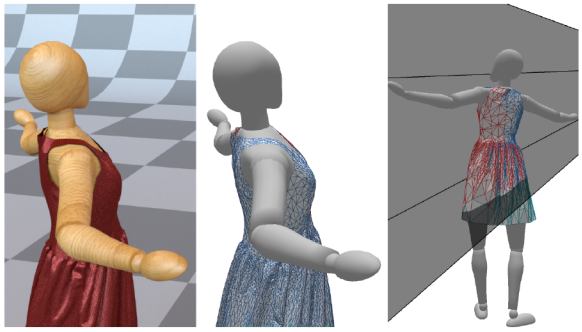

Figure 7: *A solo dancer is shown from the camera's perspective (left, middle) and from an external perspective with the camera's view frustum drawn in black wireframe (right).*

3.4. Interpenetration handling

Remeshing invariably produces changes in the geometry of the cloth mesh, and can introduce interpenetrations of the cloth with itself or with obstacles. We found that the simple approach for interpenetration handling used in previous remeshing work [NSO12] does not always converge to an interpenetration-free configuration in the presence of the agressive coarsening we perform in non-visible regions. Instead we use the approach of intersection contour minimization proposed by Volino et al. [VMT06], which we found to be robust to large and complex interpenetrations.

4. Results and Discussion

We have implemented the methods described in this paper as an extension to the ARCSim simulation code. These extensions will be integrated into the v0.3.0 distribution.

To test the utility of the proposed methods, we ran comparisons with adaptive and non-adaptive simulations on several

	Num. Faces			Num. Vertices			Time / Frame (seconds)			Speed-up
	min	max	mean	min	max	mean	min	max	mean	
Karate										
Non-adaptive	123,790	123,790	123,790	63,889	63,889	63,889	136.47	288.01	161.44	0.49×
Adaptive	25,785	71,142	41,973	14,135	37,199	22,358	31.41	184.39	79.04	1×
View-dependent	5,223	39,139	21,501	3,142	20,770	11,670	5.52	83.29	33.02	2.39×
Solo Dancer										
Non-adaptive	43,710	43,710	43,710	22,736	22,736	22,736	27.31	58.09	29.27	0.54×
Adaptive	12,030	21,763	18,041	6,593	11,535	9,659	7.78	22.00	15.80	1×
View-dependent	730	15,951	9,638	560	8,599	5,314	0.83	13.81	7.55	2.09×
Multiple Dancers										
Non-adaptive	437,100	437,100	437,100	227,360	227,360	227,360	273.13	580.85	292.73	0.47×
Adaptive	119,515	184,340	161,028	65,505	98,630	86,897	79.97	178.24	136.99	1×
View-dependent	11,216	102,945	36,339	7,619	56,285	21,228	12.02	82.54	30.76	4.45×

Table 1: *Statistics and timing numbers for the examples. Non-adaptive simulations use a fixed high-resolution mesh. Adaptive simulations use the unmodified scheme implemented in ARCSim. View-dependent simulations use the method described in this paper. The adaptive simulations are used as a baseline for comparison. The non-adaptive mesh resolution was selected to match the visual quality of the adaptive simulations.*

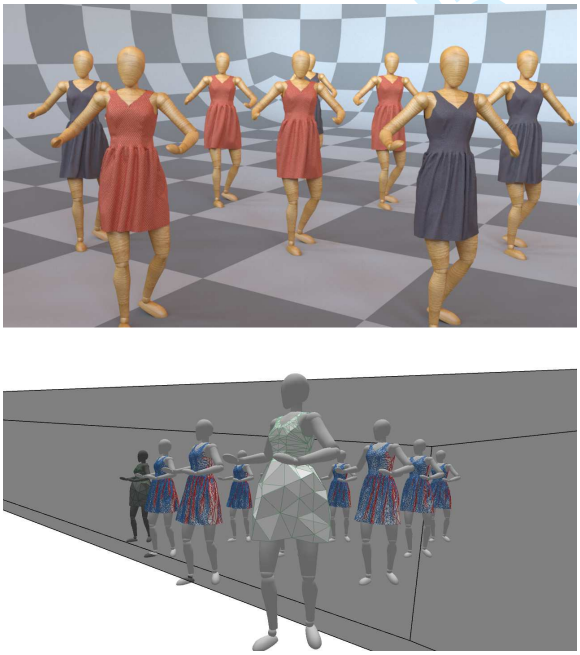

Figure 8: *A group of multiple dancers is rendered from the camera's perspective (top), while an external view showing the camera's view frustum is shown below. Characters completely outside the view frustum are simulated at very low resolution.*

examples. All simulations were run on a machine with an Intel Xeon E5-2680 v2 processor and 16 GB RAM using 4 execution threads. The constants for view-dependent refinement that were used for these examples are $v_{\text{back}} = 0.2$, $m = 0.4$ m, $T = 5$ frames, and $v_{\text{out}} = 0.01$. For the example shown in Figure 1, we used larger $v_{\text{out}} = 0.1$ to avoid intersection problems with the layered garments.

Figure 1 shows a character wearing a layered outfit consisting of three interacting garments. As the accompanying video shows, the character's clothing exhibits complex dynamic motion with detailed wrinkles. As the camera moves continuously or transits between cuts, the simulation mesh is updated to maintain a constant level of visible detail while out-of-view regions are aggressively coarsened. Figure 6 plots the time per frame and total cumulative computation time for this example for both the basic adaptive simulation and our view-dependent version.

Figure 7 shows a character wearing a simple dress, showing a degree of view-dependent refinement and coarsening similar to Figure 1. A group of ten such characters is shown in Figure 8. Note that while these characters are all performing the same actions in the same type of dress, they are simulated individually with different resolutions depending on their location relative to the viewpoint. In practical applications, multiple characters in a crowd would have different garments and motions and would therefore also have to be simulated individually even without view dependence.

Timings for these three examples are reported in Table 1. The single character examples realize a speed-up between 2.1× and 2.4×. This speed-up becomes more substantial for the group of ten characters where it is roughly 4.5×. The greater speed-up occurs because when a single character fills the screen, requiring full adaptive resolution, it tends to obscure others which then switch to low resolution and generate substantial savings. In order for all characters to be visible, the camera must be fairly far back which yields savings across all characters. These same factors would hold

more strongly for larger groups. Accordingly we hypothesize that massive scenes involving thousands of characters would likely realize speed-ups substantially larger than those we have observed for small groups.

Remeshing does introduce small amounts of motion that might be visible if the character is otherwise still. As shown in the video, however, this motion does not manifest as visible popping because the dramatic remeshing generally happens off screen. The motion that appears on screen is only gentle swaying that looks fairly natural. If the subject is moving, then this gentle motion is completely masked by the normal motion of the cloth. Even with a still subject, the artificial motion can still be hard to detect due to the movement of the camera.

5. Conclusions

The methods we have described provide a simple way of achieving computational savings for cloth simulation. For scenes involving multiple characters or large crowds, these savings can be substantial. We have demonstrated our approach in simulations of clothing, but believe that it could equally well be applied to other objects that can be simulated in an adaptive framework, including materials that can crumple [NPO13] or fracture [PNdJO14].

The main limitation of our method is that it requires an adaptive framework. However, once that framework is in place, view-dependent adaptivity is relatively simple to implement. For the group of dancers, our view-dependent adaptive simulation is nearly 10× faster than an equivalent non-adaptive simulation. We believe that such large performance gains outweigh any costs associated with changing mesh topology. We also believe that our approach would scale very well to massive scenes with thousands of actors, where it would produce even larger savings.

In non-visible regions that are heavily coarsened, we sometimes see a ballooning effect (for example in Figure 3) due to artificial stiffening: the wrinkles that would normally allow the cloth to hang loosely cannot be represented on such a coarse mesh. While our spatial and temporal smoothing prevent this effect from causing visible artifacts in our examples, it could pose a challenge in scenes with extremely long, flowing sheets where the motion of the visible part of the sheet might be strongly affected by parts off screen. A potential strategy to avoid this artificial stiffening would be to switch coarse faces to a material model that permits compression in lieu of sub-element buckling [CK02, WOR10].

As mentioned in the introduction, our method targets high-quality offline simulation, and there would be additional challenges in applying it to interactive animation. In particular, our temporal smoothing requires a prescribed camera motion, so it may not be possible to apply it in interactive settings where camera motions are not predetermined.

In general, simulations used for physically based animation in graphics have been designed so that they capture visible phenomena for realistic appearance. These simulations typically avoid the type of error analysis that one finds in most engineering contexts because it is difficult to quantify a measure of perceptual error that would be relevant to graphics simulations. The work we've presented here explicitly drives adaption according to a heuristic measure of visible error. Future work in this area could more explicitly take perceptual error into account so that reduced resolution could be used where masking effects due to motion, complexity, or other phenomena cause humans to be less sensitive to apparent detail.

Acknowledgments

We thank the other members of the Visual Computing Lab at UC Berkeley for their valuable advice and help. This work was supported by the Intel Science and Technology Center for Visual Computing (ISTC-VC), an AWS in Education Grant award, and a gift from Pixar Animation Studios. Woojong Koh was supported by a Samsung Scholarship.

We built our method on top of the ARCSim framework developed in UC Berkeley, and the images in this paper and video were rendered with the Mitsuba renderer by Wenzel Jakob.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[ATW13] ANDO R., THÜREY N., WOJTAN C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. 32*, 4 (July 2013), 103:1–103:10. 2

[Bar06] BARRAN B. A.: *View dependent fluid dynamics*. Master's thesis, Texas A&M University, 2006. 2

[BEB12] BROCHU T., EDWARDS E., BRIDSON R.: Efficient geometrically exact continuous collision detection. *ACM Trans. Graph. 31*, 4 (July 2012), 96:1–96:7. 2

[BK04] BEAUDOIN J., KEYSER J.: Simulation levels of detail for plant motion. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), SCA '04, pp. 297–304. 2

[BK11] BUNLUTANGTUM R., KANONGCHAIYOS P.: Enhanced view-dependent adaptive grid refinement for animating fluids. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry* (2011), VRCAI '11, pp. 415–418. 2

[CFL*07] CHENTANEZ N., FELDMAN B. E., LABELLE F., O'BRIEN J. F., SHEWCHUK J. R.: Liquid simulation on lattice-based tetrahedral meshes. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2007* (Aug. 2007), pp. 219–228. 2

[CH97] CARLSON D. A., HODGINS J. K.: Simulation levels of detail for real-time animation. In *Proceedings of the Conference on Graphics Interface '97* (1997), pp. 1–8. 2

[CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. *ACM Trans. Graph. 21*, 3 (July 2002), 604–611. 7

[CM11] CHENTANEZ N., MÜLLER M.: Real-time Eulerian water simulation using a restricted tall cell grid. In *ACM SIGGRAPH 2011 Papers* (2011), SIGGRAPH 2011, pp. 82:1–82:10. 2

[DWS*97] DUCHAINEAU M., WOLINSKY M., SIGETI D. E., MILLER M. C., ALDRICH C., MINEEV-WEINSTEIN M. B.: ROAMing terrain: Real-time optimally adapting meshes. In *Proceedings of the 8th Conference on Visualization '97* (1997), VIS '97, pp. 81–88. 2

[FS93] FUNKHOUSER T. A., SÉQUIN C. H.: Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH '93, pp. 247–254. 2

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A simple framework for adaptive simulation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (2002), SIGGRAPH 2002, pp. 281–290. 2

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (1997), SIGGRAPH '97, pp. 189–198. 2

[Hop98] HOPPE H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings of the Conference on Visualization '98* (1998), VIS '98, pp. 35–42. 2

[HPH96] HUTCHINSON D., PRESTON M., HEWITT T.: Adaptive refinement for mass/spring simulations. In *7th Eurographics Workshop on Animation and Simulation* (1996), Springer-Verlag, pp. 31–45. 2

[KFCO06] KLINGNER B. M., FELDMAN B. E., CHENTANEZ N., O'BRIEN J. F.: Fluid animation with dynamic meshes. In *Proceedings of ACM SIGGRAPH 2006* (Aug. 2006), pp. 820–825. 2

[KIC06] KIM J., IHM I., CHA D.: View-dependent adaptive animation of liquids. *ETRI Journal 28*, 6 (Dec. 2006), 697–708. 2

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 457–462. 2

[LV05] LI L., VOLKOV V.: Cloth animation with adaptively refined meshes. In *Proc. 28th Australasian Computer Science Conference* (2005), vol. 38. 2

[NPO13] NARAIN R., PFAFF T., O'BRIEN J. F.: Folding and crumpling adaptive sheets. *ACM Trans. Graph. 32*, 4 (July 2013), 51:1–51:8. 2, 7

[NSO12] NARAIN R., SAMII A., O'BRIEN J. F.: Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 147:1–10. Proceedings of ACM SIGGRAPH Asia 2012, Singapore. 2, 4, 5

[OFL01] O'BRIEN D., FISHER S., LIN M.: Automatic simplification of particle system dynamics. In *The Fourteenth Conference on Computer Animation. Proceedings* (2001), pp. 210–257. 2

[PNdJO14] PFAFF T., NARAIN R., DE JOYA J. M., O'BRIEN J. F.: Adaptive tearing and cracking of thin sheets. *ACM Trans. Graph.* (2014). To appear. 2, 7

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (2005). 2

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Trans. Graph. 30*, 4 (July 2011), 81:1–81:8. 2

[TC90] THINGVOLD J. A., COHEN E.: Physical modeling with B-spline surfaces for interactive design and animation. *SIGGRAPH Comput. Graph. 24*, 2 (Feb. 1990), 129–137. 2

[VB05] VILLARD J., BOROUCHAKI H.: Adaptive meshing for cloth animation. *Engineering with Computers 20*, 4 (2005), 333–341. 2

[VMT06] VOLINO P., MAGNENAT-THALMANN N.: Resolving surface collisions through intersection contour minimization. *ACM Trans. Graph. 25*, 3 (July 2006), 1154–1159. 5

[WLL*03] WARD K., LIN M. C., LEE J., FISHER S., MACRI D.: Modeling hair using level-of-detail representations. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)* (2003), CASA '03, pp. 41–47. 2

[WOR10] WANG H., O'BRIEN J., RAMAMOORTHI R.: Multi-resolution isotropic strain limiting. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 156:1–156:10. 7

[XESV97] XIA J. C., EL-SANA J., VARSHNEY A.: Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (Apr. 1997), 171–183. 2