View-Dependent Adaptive Cloth Simulation

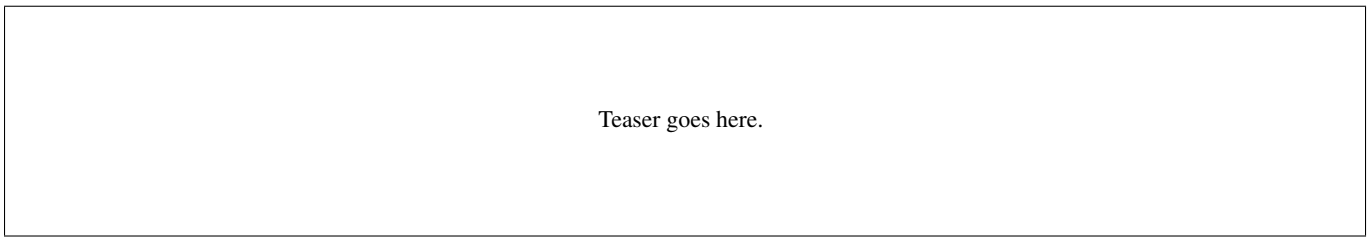

Figure 1: *Teaser caption goes here.*

Abstract

Abstract goes here.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation.

Keywords: Keywords go here.

Links: DL PDF

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, most 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 motion 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, and the cloth motion must appear coherent throughout. In closeup shots where only part of a character is in-frame, savings could be realized by reducing the computation devoted to out-of-frame cloth, so long as such savings don't 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 are viewed 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 less refined.

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.

Our work builds on the publicly available ARCSim framework 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 metric used by ARCSim so that local mesh visibility is accounted for during refinement.

In cases where a only single character is being modeled, we realize modest savings of roughly a ¡¿× speedup due to coarsening out-of-view and back-facing parts of the character's clothing. For small crowd scenes the savings are substantially larger, up to ¡¿×, as background and out-of-view characters are automatically coarsened. For massive crowd scenes with thousands of agents, we expect even greater savings would be realized.

2 Related work

3 Methods

Our view-dependent adaptive remeshing scheme is built based on the adaptive anisotropic remeshing scheme described by Narain et al. [2012]. We introduced a new view-dependent refinement in addition to their dynamical and geometrical refinements, and got a significant amount of speed-up by extending the application domain of adaptive remeshing from physical properties to perceptual properties as well.

For completeness, we explain briefly the adaptive anisotropic remeshing scheme [Narain et al. 2012]. First of all, the main refinements are materialized by the 2×2 symmetric tensor field $\mathbf{M}$. They call it *sizing field* because it determines the maximum allowable edge length under consideration of all the dynamical criteria, such as local curvature, velocity gradient, compressive strain and obstacle proximity.

Though the sizing field governs an edge length, the field itself is defined per face, not per vertex, because most of the dynamical criteria

are defined using the finite element basis functions of faces, and also anisotropy is only meaningful in terms of a face. Thus, after computing the sizing fields of each face, we need to recalculate another sizing field $\mathbf{M}$ of a vertex by weighted-averaging the original sizing fields of all adjacent faces, which can finally determine whether incident edges are valid or not in combination with the sizing field of the other end as follows: If

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

when $\mathbf{u}_{ij}$ is a vector in the parameter space representing the edge between ith and jth vertices, the edge satisfies all the dynamical criteria. If not, the edge is labeled as invalid, and we split the edge to make it not to violate the maximum edge length defined by the dynamical criteria.

In overall, the scheme first refine the mesh by splitting until there is no invalid edge, then coarsen the mesh by collapsing edges as long as not introducing a new invalid edge, all with respect to the sizing field $\mathbf{M}$.

This original scheme achieved very high-quality cloth simulation, so we focused on the speed rather than the quality, and improved the simulation speed by optimizing for one of the frequent cases: running simulation with a given camera motion.

As we mentioned in the introduction, from the camera's point of view, a part of triangle meshes that is out of frame, facing away from the camera, or at a distance will be less important than others, and we can save the computational effort being wasted by reducing the resolution of the simulation mesh partially on those parts without introducing any noticeable artifacts.

Specifically, we present the method that manipulates the sizing field $\mathbf{M}$ for each face so that the size of each face is no more than what we need to achieve the visually equivalent simulation. By changing the resolution of the simulation mesh adaptively, we can achieve faster simulation speed than the previous method neither losing any significant visual detail nor introducing any noticeable artifacts.

Two objectives. Assume that all the adjacent faces of vertices i and j are non-visible if considering the current camera position, and, on the other hand, all the adjacent faces of vertices k and l are visible. Then, our objectives for view-dependent remeshing are roughly two like the following:

$$s(i,j)^2 > s_{\text{vd}}(i,j)^2, \tag{2}$$
$$\|\mathbf{S}\mathbf{u}_{kl}\| \geq c, \tag{3}$$

where $s_{\text{vd}}(\cdot)$ is a sizing function with our view-dependent criteria, $\mathbf{S}$ is a transformation matrix from material space to screen space, and c is some multiples of the pixel size in screen space.

We need to do uniform scaling the sizing field for the first objective, and limiting spectral radius of the sizing field for the second. Each component of our algorithm is related with one of these objectives, and the components are as follows: (1) view factor F, a scalar field for sizing field tensors, (2) spatial smoothing, (3) temporal anticipation (4) screen-space resizing, (5) silhouette preserving, and (6) interpenetration resolving.

3.1 Multiplying sizing field by scalar

We call this scalar *view factor*, and just scale the sizing field $\mathbf{M}$ by the square of it in order to directly change the size of face. Please note that, due to the inequality condition for the invalid edge (Ineq. 1), the larger sizing field actually means the smaller edge length and vice versa. The sizing field (per face) scaled by the square of view factor F is as follows:

$$\mathbf{M}_{\text{vd}} = F^2\mathbf{M} \tag{4}$$
$$= \frac{\mathbf{M}_{\text{crv}}}{(\Delta n_{\max}/F)^2} + \frac{\mathbf{M}_{\text{cmp}}}{(c_{\max}/F)^2} + \frac{\mathbf{M}_{\text{vel}}}{(\Delta v_{\max}/F)^2} + \frac{\mathbf{M}_{\text{obs}}}{F^{-2}}. \tag{5}$$

We can interpret the physical meaning of uniform scaling. Multiplying the sizing field tensor $\mathbf{M}$ by a scalar is actually adjusting maximum allowable changes in vertex normal, material compression, and velocity difference across any edge. As for the obstacle sizing tensor, it virtually increases a distance from a vertex to an obstacle's tangent plane by factor of $1/F$ because the definition of $\mathbf{M}_{\text{obs}}$ is

$$\mathbf{M}_{\text{obs}}^i = \frac{\nabla\varphi_i \nabla\varphi_i^T}{\varphi_i(\mathbf{u}_i)^2}, \tag{6}$$

where φ is a signed distance of a vertex to an obstacle's tangent plane.

Basic visibility. With camera trajectory and parameters, such as field of view and near/far clipping distances, given, we can compute a viewing frustum at any given time. Thus, if a face is inside of the volume at current time, then we will give view factor of 1 to the face, and if not, we will give view factor of F_{out} to the face. Similarly, even though a face is inside of the viewing frustum, if the face is facing backward, then we will also set F_{out} to it. That is our basic view-dependent criteria based on simple visibility check, and from now on view factor F is a scalar function of face, not a constant, as follows:

$$F_1(i) = \begin{cases} 1 & \text{if face } i \text{ is inside and forward-facing,} \\ F_{\text{out}} & \text{if face } i \text{ is outside } \textit{or} \text{ backward-facing.} \end{cases} \tag{7}$$

However, we will omit the face number i unless we need to use multiple faces at the same time in our explanation.

Spatial smoothing. As you can see, our basic criteria is too simple to guarantee artifact-free, and it is mainly due to the discontinuities in the scalar field.

First of all, you may notice that there is a large discontinuity in the scalar field F along with the boundary between in-frame and out-of-frame faces. This discontinuity can make in-frame faces moves differently than the original simulation by transferring invalid forces from the out-of-frame region. Popping is the most common artifact which is caused by too few degrees of freedom while resolving collision response.

Thus, we need a buffer zone in between in-frame and out-of-frame faces to prevent this error propagation. The buffer zone basically consists of smooth falloff of the scalar values from 1 to F_{out}. The

buffer zone, or transition zone is set by a distance from the view frustum's six faces to a face.

Let a signed distance from a view frustum to a face be d (negative if inside), and a margin length being set for smoothly transitioning from inside to outside be m. Then, we can compute a normalized distance $\delta(d)$ from a view frustum and, in turn, the smoothed view factor F_2 as follows:

$$F_2(d) = (1 - \delta(d))F_1 + \delta(d)F_{\text{out}}, \tag{8}$$

$$\delta(d) = \text{clamp}(\frac{d}{m}, [0, 1]). \tag{9}$$

Basically, F_2 is a linear interpolation between F_1 and F_{out}, and we can update the view factor in Equation 4 as follows:

$$\mathbf{M}_{\text{vd}} = F_2^2\mathbf{M} \tag{10}$$

You may see another discontinuity on the boundary in between front-facing faces and backward-facing faces, but we will handle this discontinuity later in section "Silhouette preserving".

Temporal anticipation. However, F_2 is not the final view factor we used in our implementation. We just handled discontinuity in spatial domain, which means we smoothed out a rapid change in view factor F across any edge, but there might be discontinuity in temporal domain as well. In our cases, a jump cut, which is moving a camera discontinuously from frame to frame, can ensue temporal discontinuity in view factor field.

Thus, we use a temporal anticipation to avoid discontinuous changes in view factor over time, and, in turn, discontinuous changes in the resolution of the simulation mesh, which may cause noticeable popping artifacts.

Our temporal anticipation is to set a temporal window in which we will compute several view factors in advance and take the maximum value among them. By doing so, it ensures that the cloth mesh gets refined to sufficient resolution before it appears in the view.

Please note that, if we just use the values as they are, the same problem occurs when very large or small view factor first enters the window. Therefore, we use a linear attenuation as it goes far from the current frame like the following:

$$F_3 = \max(w_m F_2(d_{t-m}), \dots, w_0 F_2(d_t), \dots, w_m F_2(d_{t+m})), \tag{11}$$

where t is the current time, d_{t+m} is a distance from the face at time t to the viewing frustum at time $t + m$, and weight $w_i = 1 - i/m$. One of its variants is to set $w = w_0$ up to the next n frames like

$$F_3' = \max(\dots, w_0 F_2(d_t), \underbrace{w_0 F_2(d_{t+1}), \dots, w_0 F_2(d_{t+n})}_{n}, w_1 F_2(d_{t+n+1}), \dots). \tag{12}$$

Finally, F_3' is the final view factor we used in our implementation. With F_3' and Equation 4, we can fix view-dependent sizing field $\mathbf{M}_{\text{vd}}$ like the following:

$$\mathbf{M}_{\text{vd}} = F_3^2\mathbf{M}. \tag{13}$$

3.2 Limiting spectral radius of sizing field

Until now, we discussed on how to scale the sizing field $\mathbf{M}$ of non-visible faces, which will change the size of its corresponding face directly. Now we need to discuss on how to optimize *visible faces*.

Our objective for the visible faces is Inequality 3: we want them to be large enough to affect the visual appearance of cloth on the image plane. In other words, we want to them at least some multiples of the pixel size in screen space.

There are two types of information we can utilize for optimizing visible faces: (1) the orientation of a face and (2) the distance from a camera to a face.

First of all, the orientation of a face. The basic idea behind this is that the projected area of a triangle become smaller as the face normal becomes perpendicular to the viewing direction. This is called foreshortening and we can enlarge a slanted face along the viewing direction with small changes of the area of a projected face.

Secondly, the distance from the camera. The farther a face is from a camera, the smaller it looks under perspective projection. Like the orientation, the distance to a camera also affects the projected area of the triangles under perspective projection.

Screen-space resizing. To take into account the effects of both orientation and the distance to a camera of visible faces in unified and consistent way, we developed screen-space resizing.

For visible faces, we don't want to change the edge length computed by the original dynamical criteria[1] directly. However, we also don't want edges too short to be seen after projected on the screen. Thus, what we want to do is to bound the shortest allowable edge length in screen space to some multiples of the pixel size.

Bounding the shortest allowable edge length is possible by limiting the spectral radius of sizing field $\mathbf{M}$. This is already used in the original scheme [Narain et al. 2012] to bound the minimum allowable edge length in material space.

Along the same line, we first transform the $\mathbf{M}_{\text{vd}}$ from material space to screen space, and bound the shortest allowable edge length in screen space, and transform it back to material space like the following:

$$\tilde{\mathbf{M}}_{\text{vd}} = \mathbf{S}^T\mathbf{M}_{\text{vd}}\mathbf{S}, \tag{14}$$

$$\hat{\mathbf{M}}_{\text{vd}} = \mathbf{S}^{-T}L(\tilde{\mathbf{M}}_{\text{vd}})\mathbf{S}^{-1}, \tag{15}$$

$$\text{s.t. } \mathbf{S} = \mathbf{N}\tilde{\mathbf{S}}, \tag{16}$$

where $\mathbf{N}$ is a nonuniform scale matrix for aspect ratio, $\tilde{\mathbf{S}}$ is the Jacobian matrix of the screen space with respect to the material space, which is discretized by finite element basis functions on each face[2], and $L(\cdot)$ is a function limiting the spectral radius of sizing field.

What function $L(\cdot)$ does is like the following. Let the shortest allowable edge length be $\ell_{\text{min,ss}} = p/h$, where p is the minimum number of pixels for edge in screen space and h is the vertical resolution of the screen (because the aspect ratio matrix converts the screen height to 1), and let the eigendecomposition of the sizing field be $\tilde{\mathbf{M}}_{\text{vd}} = \mathbf{Q}\hat{\mathbf{\Lambda}}\mathbf{Q}^T$ with eigenvalues $\hat{\lambda}_1, \hat{\lambda}_2$.

[1] The view factor for visible faces is always 1. See Equation 7.

[2] Please note that the transformation matrix $\tilde{\mathbf{S}}$ should be computed per face because it uses finite element basis functions defined on each face.

Then, we can directly set the minimum allowable edge length in screen space to $\ell_{\text{min,ss}}$ as follows:

$$\lambda_i = \min(\hat{\lambda}_i, \ell_{\text{min,ss}}^{-2}), \tag{17}$$

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2), \tag{18}$$

$$\hat{\mathbf{M}}_{\text{vd}} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T. \tag{19}$$

The final sizing field $\hat{\mathbf{M}}_{\text{vd}}$ is what the function $L(\cdot)$ returns at the end.

Hard limit on spectral radius. After all, there is a hard limit on edge length. For example, even if a face is infinitely far away from a camera and is resized by screen-space resizing, the size of the face cannot be infinitely large because there is a code limiting to the maximum allowable length at the end of our algorithm's every step, which is similar to the geometrical criteria of the original scheme [Narain et al. 2012].

3.3 Transferring sizing field from face to vertex

If not mentioned otherwise, every sizing field $\mathbf{M}$ we mentioned until now is per-face sizing field. Now we need to transfer per-face sizing fields to per-vertex sizing fields to make ARCSim working. We basically use the same approach in the paper [Narain et al. 2012], but we use a different approach in some particular cases to help our discontinuity problem.

Silhouette preserving. As we mentioned earlier, still one of spatial discontinuities is left untreated: the boundary between forward-facing and backward-facing faces. In order to handle this discontinuity on view factors, we switch our averaging scheme to more conservative one adaptively.

In the paper by Narain et al. [2012], they used the area weighted averaging for computing per-vertex sizing field from per-face sizing fields of adjacent faces. We use the same approach as long as $\partial F/\partial \mathbf{u}$ is less than some threshold. Otherwise, we use a different approach that computes a new sizing tensor that encloses all the adjacent sizing tensors and assign it to the vertex [Narain et al. 2013].

This approach basically preserves the shortest edge length on the boundary. Thus, it will make the silhouette of cloth conforming to the visible side of cloth, which has higher resolution than the non-visible side in general. Also, it smooth out a small number of backward-facing faces temporarily appearing in the middle of the visible side of the cloth, such as small wrinkles.

Since in our cases this will happen mainly on the boundary between forward-facing and backward-facing faces and the scheme makes the silhouette looks smoother, we call it *silhouette preserving*.

3.4 Misc.

Interpenetration handling. We newly implemented surface collision resolving algorithm by Volino and Magnenat-Thalmann [2006] instead of the original interpenetration handling algorithm by Narain et al. [2012].

4 Results and Discussion

Results need to go here!

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

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 which would be relevant to graphics simulations. The work we've presented here explicitly drives adaption based on a heuristic measure of visible error. We believe that future work in this area could more explicitly take perceptual error into account so that reduced resolution could be used were masking effects due to motion, complexity, or other phenomena cause humans to be less sensitive to apparent detail.

References

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

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

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