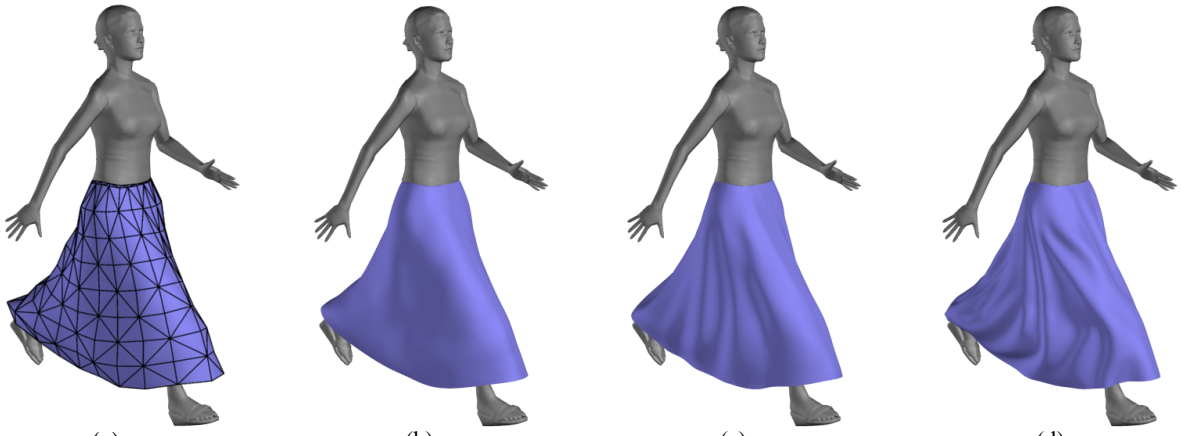
Physics-Inspired Upsampling for Cloth Simulation in Games

Ladislav Kavan*
Disney Interactive Studios

Dan Gerszewski
Disney Interactive Studios
University of Utah

Adam W. Bargteil
University of Utah

Peter-Pike Sloan
Disney Interactive Studios



(a) (b) (c) (d)

Figure 1: *(a) Coarse simulation, (b) subdivision, (c) our proposed upsampling and (d) fine-scale simulation. Our upsampling operator is learned from a small set of coarse and fine-scale examples, which allows it to achieve higher quality than subdivision while still being linear and therefore very efficient and simple to implement (this example is upsampled in 0.8ms on a single CPU thread).*

Abstract

We propose a method for learning linear upsampling operators for physically-based cloth simulation, allowing us to enrich coarse meshes with mid-scale details in minimal time and memory budgets, as required in computer games. In contrast to classical subdivision schemes, our operators adapt to a specific context (e.g. a flag flapping in the wind or a skirt worn by a character), which allows them to achieve higher detail. Our method starts by pre-computing a pair of coarse and fine training simulations aligned with *tracking constraints* using harmonic test functions. Next, we train the upsampling operators with a new regularization method that enables us to learn mid-scale details without overfitting. We demonstrate generalizability to unseen conditions such as different wind velocities or novel character motions. Finally, we discuss how to re-introduce high frequency details not explainable by the coarse mesh alone using *oscillatory modes*.

CR Categories: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

Keywords: Cloth simulation, data-driven animation, upsampling, video games.

Links: DL PDF WEB VIDEO

*e-mail: ladislav.kavan@gmail.com

1 Introduction

Cloth simulation has become commonplace in computer generated movies, and is slowly but surely finding its way into computer games, with run-time solutions available commercially from NVIDIA PhysX™and Havok™and as open-source from the Bullet Physics Library. One challenge is that current games are complex pieces of software executing many inter-dependent tasks, including rendering, animation, artificial intelligence, gameplay, human-computer interaction and networking, with frame budgets of 16-33ms. Because advanced effects such as cloth are typically not vital components of a game, the time budget for most developers is around 1ms. With commodity CPUs, this time budget only allows very coarse simulation meshes, inadequate for direct display. While the computing power of modern GPUs is sufficient to simulate high-resolution meshes in real-time, many games choose to spend the majority of their GPU budgets on rendering. In the future we can expect more powerful hardware, however, light-weight solutions will always be important for the increasingly popular, low-power, mobile devices.

Many games often resort to pre-computed solutions with limited flexibility [Herman 2001; Kavan et al. 2010] or subdivided coarse simulation with limited detail. Subdivision has a long history in computer graphics and is frequently applied to cloth. The most common subdivision schemes are linear and feature very efficient implementations [Loop 1987]. Recent work on adding detail to coarse simulations departs from the linear schemes and focuses on high-resolution detail synthesis using advanced non-linear operators [Feng et al. 2010; Rohmer et al. 2010], simplified fine-scale physics [Müller and Chentanez 2010] or comprehensive databases of example shapes [Wang et al. 2010a]. While real-time results have been demonstrated using high-end graphics hardware, the current gaming market is dominated by consoles, which have far more limited computing resources.

In this paper, we focus on *linear* upsampling operators that offer very simple and efficient implementations across a number of platforms. We aim at delivering interesting mid-scale details missing in the

coarse scale simulation, while keeping our method general and supporting both character clothing and environmental cloth. In contrast to subdivision, we consider dense upsampling matrices that are specialized for a given context, such as a skirt worn by a character or a flag flapping in the wind, which allows us to trade-off complete generality for higher visual quality. This trade-off is justified because in games interaction possibilities are often limited by design (e.g., the flag will not be removed from the pole and a skirt will always be attached at the waist). At the same time, we allow for rich interactivity in the chosen subspace (e.g., changing wind velocity or applying unexpected motions), making our approach more responsive than pre-computed solutions.

Our main contribution is a method for designing linear upsampling operators that capture context-specific mid-scale details using a small set of training data, such as a skirt animated in a walking motion or a flag flapping in the wind with constant speed. We do not attempt to exhaustively sample the set of all possible animations, which would be hard or even impossible in modern games. Instead, we propose *harmonic regularization* (Section 3), a regularization method that extends the well-known Tikhonov approach and allows for precise control between capturing details and overfitting. After a one time mesh-dependent pre-computation, our fitting process is fast enough for interactive authoring, enabling us to put an artist in the loop and facilitate deployment to production. To generate high-quality training data, we align the coarse and fine-scale cloth meshes using tracking constraints [Bergou et al. 2007], proposing *harmonic test functions* (Section 4). Finally, we discuss a method to re-introduce fine-scale traveling waves (if any) using *oscillatory modes*, useful, for example, in simulations subjected to strong wind (Section 5). Our prototype implementation achieves speeds around 1ms on a single CPU core for both coarse simulation and upsampling.

2 Related Work

Cloth simulation. We only scratch the surface of the physically based animation literature, please refer to the surveys [Nealen et al. 2005; Müller et al. 2008] for a more comprehensive introduction. Past decades witnessed important developments in collision handling [Bridson et al. 2002], cloth energy models [Grinspun et al. 2003], time integration [Baraff and Witkin 1998; Harmon et al. 2009] and cloth inextensibility [Goldenthal et al. 2007; English and Bridson 2008]. Recently, position-based approaches to dynamics [Jakobsen 2001; Müller et al. 2007; Müller 2008] are becoming increasingly popular in the computer graphics industry [Stam 2009]. The closely related idea of strain limiting [Provot 1995] also remains an active research area [Thomaszewski et al. 2009; Wang et al. 2010b]. Our approach is not tied to a specific simulation paradigm, but for our experiments we chose the position based dynamics of Müller and colleagues [2007] for its speed and stability. The TRACKS framework [Bergou et al. 2007] was developed to address the art-directability problem of cloth simulations by constraining the fine-scale "tracked" simulation to match a given coarse "guide" animation; related techniques have also been studied for deformable solids [Barbič and Popović 2008; Barbič et al. 2009]. In this paper, we employ tracking to generate well-behaved training data.

Subdivision. Subdivision surfaces [Zorin et al. 2000] provide the most common way to obtain smooth rendering meshes from coarse simulation data. Their application to cloth and character animation is classic [DeRose et al. 1998], with a popular method for triangular meshes due to Loop [1987]. For rendering meshes with arbitrary connectivity, similar results can be obtained using geometry-aware bases [Sorkine et al. 2005]. However, these methods only guarantee geometric smoothness and have no information about material properties or external forces. Our approach strives to extend these concepts to physically inspired details while maintaining linearity.

Procedural wrinkle synthesis. Coarse simulation meshes can also be enriched with details using physically based buckling models, typically resulting in nonlinear methods assuming (quasi-)isometry of the fine mesh [Hadap et al. 1999; Kang et al. 2001; Larboulette and Cani 2004; Tsiknis 2006; Loviscach 2006; Kang and Lee 2007]. The latest procedural techniques [Rohmer et al. 2010] trade real-time performance for realistically animated wrinkles in any mesh configuration. Müller and Chentanez [2010] synthesize very detailed wrinkles using a simplified dynamics solver running on the GPU. Procedural wrinkle synthesis can be layered on top of our method if greater detail is desired; for fine-scale traveling waves, one option is to apply our oscillatory modes (Section 5).

Data driven methods. An important alternative to physically based simulation is data-driven modeling of deformable objects [James and Fatahalian 2003], which has been investigated for cloth both in real-time [Cordier and Magnenat-Thalmann 2005] as well as feature animation applications [Cutler et al. 2005; Kim and Vendrovsky 2008]. However, the problem continues to attract research efforts. Recently, Wang and colleagues [2010a] emphasize quality and pre-compute an extensive database of wrinkled clothing meshes for uniformly sampled joint rotations. Our method does not aim for as high a level of detail but, is orders of magnitude faster and handles loose clothing and environmental cloth. Emphasizing performance, similarly to our method, *stable spaces* [de Aguiar et al. 2010] synthesize character clothing based on body motion and cloth history. While very fast, stable spaces do not have a run-time physics component and rely on learning dynamics from the training data. Our approach has similar time and memory complexity but requires much less training data, features physically-based dynamics and is applicable to non-character clothing.

Feng and colleagues [2010] present a data-driven deformation transformer to add details to coarse simulations. Because their approach assumes only implicit alignment of coarse and fine training simulations (see Figure 2 left), the deformation transformer needs to learn how to compensate for the often highly nonlinear mismatch between the coarse and fine scale physics, greatly complicating the ability to generalize (e.g., note the difference in local orientations). While this problem is challenging, Feng and colleagues [2010] achieve promising results by combining skinning, rotational regression, kernel canonical correlation analysis [Feng et al. 2008] and clustered principal component analysis. In our approach, we avoid these problems by aligning the training data using tracking (see Figure 2 right), allowing us to achieve an order of magnitude faster run-time, much simpler implementation and support of environmental cloth.

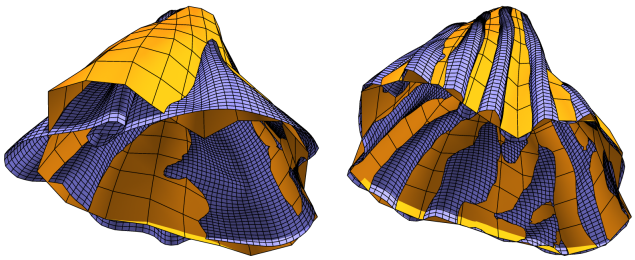


Figure 2: *A coarse (orange) and corresponding fine-scale simulation frame (purple) relying on alignment via collision objects only (left) and using tracking (right, 80 test functions).*

Regularization. Regularization is a method for solving ill-posed inverse problems by injecting additional assumptions (priors), with a popular method due to Tikhonov [Pighin and Lewis 2007]. Geometry-aware bases introduced by Sorkine and colleagues [2005] can be considered as an application of Tikhonov regularization to mesh processing, as discussed in detail by Volodine and colleagues

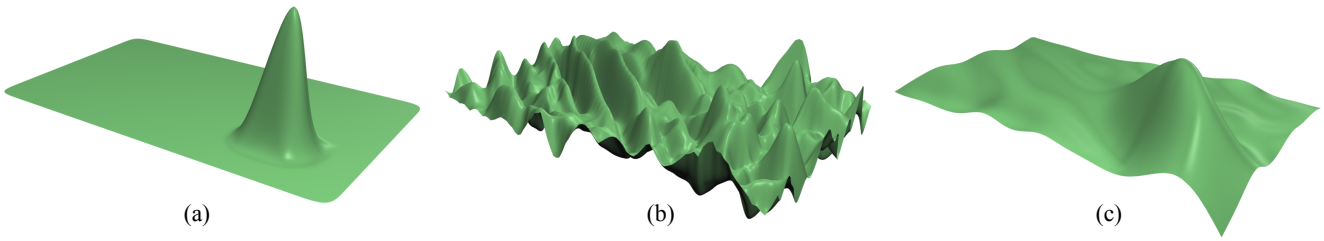


Figure 3: *An example of one column (shape function) for three upsampling operators on a flat rectangular mesh: (a) Loop subdivision, (b) non-regularized least squares fit of training data and (c) our method.*

[2006]. These methods rely on discrete Laplace-Beltrami operators [Pinkall and Polthier 1993; Wardetzky et al. 2007; Lévy and Zhang 2010] to express a smoothness prior. Using spectral techniques, we extend the concept of Tikhonov regularization to the spatio-temporal domain proposing, essentially, *animation*-aware functions. Related approaches in physics include modal decomposition and subspace methods [James and Pai 2002; Barbič and James 2005].

3 Learning Upsampling Operators

We start by reviewing the basic theory and introducing our notation. A linear upsampling operator mapping coarse points $\mathbf{p}_c \in \mathbb{R}^{M\times 3}$ to fine points $\mathbf{p}_f \in \mathbb{R}^{N\times 3}$ can be represented by a matrix $\mathbf{U} \in \mathbb{R}^{N\times M}$. In our intended applications, M is around 100-200, with N on the order of 5k-10k. The upsampling can be expressed using matrix multiplication

$$\mathbf{p}_f = \mathbf{U}\mathbf{p}_c \tag{1}$$

Note that this formulation is rotationally invariant, because for an arbitrary $\mathbf{A} \in \mathbb{R}^{3\times 3}$

$$\mathbf{U}(\mathbf{p}_c\mathbf{A}) = (\mathbf{U}\mathbf{p}_c)\mathbf{A} = \mathbf{p}_f\mathbf{A},$$

i.e., transformation by $\mathbf{A}$ followed by upsampling is equivalent to upsampling followed by the transformation. To guarantee affine invariance it is sufficient to require that the rows of $\mathbf{U}$ sum to 1. For brevity, we denote the set of matrices that satisfy the partition of unity constraint as $\mathcal{U}_1 = \{\mathbf{U} \in \mathbb{R}^{N\times M} : \sum_j U_{i,j} = 1 \; \forall i = 1,\ldots,N\}$.

Intuitively, Equation (1) takes linear combinations of the columns of $\mathbf{U}$, and therefore, spatial smoothness of the columns is critical for a smooth result. We informally call the columns of an upsampling operator *shape functions* and to visualize them we use a flat rectangular mesh. Each column corresponds to one coarse vertex, typically collocated with the maximum. For example, one of the shape functions of a Loop subdivision matrix is shown in Figure 3a (the remaining ones being just translated copies of the "bump").

3.1 Data Term

Assume we have a set of aligned training pairs $(\mathbf{p}_{c,i}, \mathbf{p}_{f,i})$ for $i = 1,\ldots,F$, where F is the number of input frames. Stacking both coarse and fine examples into matrices $\mathbf{P}_c \in \mathbb{R}^{M\times 3F}$ and $\mathbf{P}_f \in \mathbb{R}^{N\times 3F}$ allows us to write the regression problem as

$$\underset{\mathbf{U}\in\mathcal{U}_1}{\operatorname{argmin}} \|\mathbf{U}\mathbf{P}_c - \mathbf{P}_f\|, \tag{2}$$

where $\|.\|$ denotes the Frobenius norm. This is a constrained least squares problem that can be solved efficiently using standard numerical methods [Lawson and Hanson 1974]. However, even with enough training data so that $\mathbf{P}_c$ is numerically well conditioned (i.e., an overconstrained problem), this approach often results in shape functions that overfit, as can be seen in Figure 3b. While optimally reconstructing the input data, even a small perturbation away from the training frames can lead to very non-smooth results.

3.2 Harmonic Regularization

To obtain smooth upsampling even for configurations away from the training data, we introduce a smoothing term employing a symmetric regularization matrix $\mathbf{R} \in \mathbb{R}^{N\times N}$

$$\underset{\mathbf{U}\in\mathcal{U}_1}{\operatorname{argmin}} \|\mathbf{U}\mathbf{P}_c - \mathbf{P}_f\| + \|\mathbf{R}\mathbf{U}\| \tag{3}$$

We also experimented with a $\|.\|_1$-norm of the regularization term (sum of absolute values) instead of the Frobenius norm (sum of squares). The absolute values metric produces sparse, but, unfortunately, not very smooth shape functions. Also, the required optimization routines are much more complex and therefore, we focus on the Tikhonov-like form in Equation (3).

It is important to note that our regularization matrix, $\mathbf{R}$ multiplies $\mathbf{U}$ from the left, acting on the individual *columns* of $\mathbf{U}$, whereas a standard Tikhonov regularization term multiplies $\mathbf{U}$ from the right (i.e. $\|\mathbf{U}\mathbf{R}\|$), acting on the individual *rows* of $\mathbf{U}$. By acting on the columns of $\mathbf{U}$, the left multiplication encourages spatially smooth shape functions. With the simplest choice of regularization matrix, $\mathbf{R} := \alpha\mathbf{I}$, $\alpha \geq 0$, left and right multiplication are equivalent because matrix multiplication by $\alpha\mathbf{I}$ is commutative. This case is easy to solve numerically, because the solve for each row of $\mathbf{U}$ is independent. However, this simple regularization term does not result in sufficiently smooth shape functions, see Figure 4.

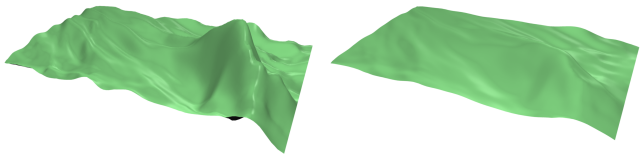


Figure 4: *Example shape function with Tikhonov regularization for $\alpha = 1$ (left) and $\alpha = 2$ (right).*

Better results can be obtained by using $\mathbf{R} := \alpha\mathbf{L}$, where $\mathbf{L} \in \mathbb{R}^{N\times N}$ is a symmetrized Laplace-Beltrami operator [Pinkall and Polthier 1993; Lévy and Zhang 2010]. Specifically, $\mathbf{L} := \mathbf{V}^{-\frac{1}{2}}\mathbf{C}\mathbf{V}^{-\frac{1}{2}}$, where $\mathbf{V} \in \mathbb{R}^{N\times N}$ is a diagonal matrix of Voronoi areas and $\mathbf{C} \in \mathbb{R}^{N\times N}$ is a sparse symmetric matrix of cotangent weights, i.e., for an edge connecting vertices i and j, $C_{i,j} = (\cot\beta_{i,j} + \cot\tilde{\beta}_{i,j})/2$ where $\beta_{i,j}$ and $\tilde{\beta}_{i,j}$ are opposite angles of the incident triangles and $C_{i,i} = -\sum_j C_{i,j}$.

However, the Laplacian regularization term $\mathbf{L}$ introduces spatial dependencies and therefore the rows can no longer be solved for independently. We can still solve for the whole $\mathbf{U}$ at once by assembling an aggregate linear system with $N \cdot M$ unknowns, but, unfortunately, the resulting system is not very sparse due to the dense data term. Even if we compress $\mathbf{P}_f$ using Singular Value Decomposition and apply a state-of-the-art direct solver, PARDISO [Schenk and Gartner 2006], the solve often takes more than an

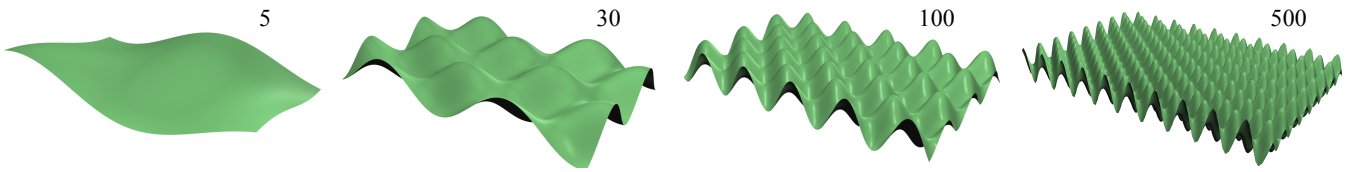


Figure 5: *Example of several low-order harmonic functions (out of a total of* 7016*) of a flat rectangular mesh. Intuitively, the harmonic functions constitute an orthonormal basis that is as smooth as possible.*

hour, which is prohibitively expensive for user tuning with different training datasets and regularization parameters (note that the system matrix depends on both $\mathbf{P}_f$ and α). We also experimented with iterative solvers such as preconditioned conjugate gradients, but found that they did not outperform the direct ones. Moreover, we observed the regularization term $\alpha\mathbf{L}$ does not prevent overfitting in *low* frequencies.

Both the time consuming numerics and the low frequency overfitting can be solved by a technique we call *harmonic regularization*. We start by factorizing $\mathbf{L} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ using the Eigendecomposition, where $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is an orthonormal matrix containing mesh harmonic functions (see Figure 5) and $\mathbf{\Lambda}$ is a diagonal matrix of Eigenvalues. This factorization allows us to rewrite the Laplacian regularization term as

$$\|\alpha\mathbf{L}\mathbf{U}\| = \alpha\|\mathbf{L}\mathbf{U}\| = \alpha\|\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{U}\| = \alpha\|\mathbf{\Lambda}\mathbf{Q}^T\mathbf{U}\|,$$

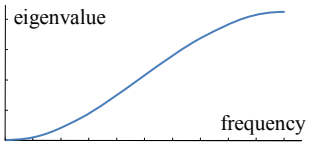
because multiplication by an orthonormal matrix does not change the Frobenius norm. Similarly, we can rewrite the data term as

$$\|\mathbf{U}\mathbf{P}_c - \mathbf{P}_f\| = \|\mathbf{Q}^T\mathbf{U}\mathbf{P}_c - \mathbf{Q}^T\mathbf{P}_f\|,$$

Substituting $\hat{\mathbf{U}} = \mathbf{Q}^T\mathbf{U}$ lets us pose the problem as

$$\underset{\hat{\mathbf{U}}\in\mathbf{Q}^T(\mathcal{U}_1)}{\operatorname{argmin}} \|\hat{\mathbf{U}}\mathbf{P}_c - \mathbf{Q}^T\mathbf{P}_f\| + \alpha\|\mathbf{\Lambda}\hat{\mathbf{U}}\| \tag{4}$$

We solve this constrained least squares problem for $\hat{\mathbf{U}} \in \mathbb{R}^{N\times M}$ and recover the upsampling operator as $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$. The advantage of this formulation is that because $\mathbf{\Lambda}$ is diagonal, the rows of $\hat{\mathbf{U}}$ can be solved for independently. Therefore, with pre-computed $\mathbf{Q}$, the solve for $\mathbf{U}$ takes only a few seconds, which makes it over two orders of magnitude faster than the direct sparse solver (see Table 1). While the factorization of $\mathbf{L}$ takes up to several minutes with our datasets, it only needs to be computed once per mesh and can then be re-used for different $\mathbf{P}_f$ and α. Alternatively, we could calculate only the low-order harmonics using the more efficient numerical techniques developed by Vallet and Lévy [2008].



Examination of the singular values $\mathbf{\Lambda}$ explains why the Laplacian smoothing term may not prevent overfitting at low frequencies—the singular values corresponding to low-order harmonics are very small (a typical spectrum is plotted in the figure to the right) and therefore these modes are not regularized enough. To remedy this situation, we propose using a regularization matrix $\mathbf{R} := \mathbf{Q}\mathbf{\Gamma}\mathbf{Q}^T$, where $\mathbf{\Gamma} \in \mathbb{R}^{N\times N}$ is a diagonal matrix generalizing the damping parameter $\alpha \in \mathbb{R}$ to N scalars, which leads to

$$\underset{\hat{\mathbf{U}}\in\mathbf{Q}^T(\mathcal{U}_1)}{\operatorname{argmin}} \|\hat{\mathbf{U}}\mathbf{P}_c - \mathbf{Q}^T\mathbf{P}_f\| + \|\mathbf{\Gamma}\hat{\mathbf{U}}\| \tag{5}$$

Due to commutativity, the simplest choice of $\mathbf{\Gamma} := \alpha\mathbf{I}$ is equivalent to Tikhonov regularization and no single α guarantees optimal results: for small α, the noisy high frequencies are not damped enough, and larger values of α quickly begin to suppress the desirable low frequencies, see Figure 4. This difficulty can be solved by specifying a different damping term for each frequency using $\mathbf{\Gamma}$.

The optimal choice of $\mathbf{\Gamma}$ depends on the characteristics of the training data and on the desired compromise between capturing details and the ability to generalize. The most straightforward way to eliminate the low frequency overfitting would be to employ $\mathbf{\Gamma} = \alpha\mathbf{I} + \beta\mathbf{\Lambda}$ or a simple step function representing a sharp cut-off at a specified frequency. However, we obtained the best results with polynomial profiles of the form $\gamma_n = a(1 + b((n-1)/N))^c$ for $n = 1, \ldots, N$ and parameters $a, b, c \geq 0$, which allow for good spectral control while smoothly attenuating high frequencies. For more details on parameter tuning please see Section 6. As an example, the shape function in Figure 3c was generated using $a = 0.8, b = 10$ and $c = 4$. Out of curiosity, we calculated the corresponding regularization matrix $\mathbf{R} := \mathbf{Q}\mathbf{\Gamma}\mathbf{Q}^T$ and found it to be almost completely dense, reinforcing our decision to move away from sparse solvers. Note that while we do not explicitly enforce localization, the smoothness prior together with the partition of unity constraint results in a noticeable spatial bias, i.e., fine vertices closer to the corresponding coarse vertex are influenced more than the ones farther away.

4 Tracking

Any data-driven approach can only produce results as good as the training data, in our case, pairs $(\mathbf{p}_{c,i}, \mathbf{p}_{f,i})$ of corresponding coarse and fine scale meshes. Even with carefully selected mesh-independent simulation parameters, the two simulations will behave differently because the finer one can represent higher frequencies; these small differences (along with numerical errors) accumulate over time and can cause the two simulations to bifurcate to different states, see Figure 2. Therefore, we choose to enforce alignment using additional constraints on the fine-scale simulation. There are several ways to formulate and enforce such *tracking constraints* [Bergou et al. 2007; Müller and Chentanez 2010] and we describe our implementation below.

Because our coarse and fine meshes represent the same surface, they are well aligned in the rest-pose, which enables us to define barycentric interpolation $\mathbf{B} \in \mathbb{R}^{N\times M}$ by projecting the fine vertices onto the coarse triangles. (We also experimented with higher order interpolation and approximation schemes but found no significant visual difference in the results.) The barycentric interpolation of the coarse mesh can therefore be written as $\mathbf{B}\mathbf{p}_c \in \mathbb{R}^{N\times 3}$. The tracking constraint requires the fine simulation state $\mathbf{p}_f$ to match certain carefully chosen degrees of freedom from $\mathbf{B}\mathbf{p}_c$ at each frame. Specifically, we define a matrix $\mathbf{T} \in \mathbb{R}^{T\times N}$ consisting of $T << N$ row vectors of mass-weighted *test functions*. Without loss of generality, we can assume the rows of $\mathbf{T}$ are orthonormal. The tracking constraint is a hard constraint on the fine simulation state $\mathbf{p}_f$ that requires

$$\mathbf{T}\mathbf{p}_f = \mathbf{T}\mathbf{B}\mathbf{p}_c \tag{6}$$

Note that since the rows of $\mathbf{B}$ form a partition of unity, the tracking constraint is invariant to simultaneous rigid body transformations of $\mathbf{p}_f$ and $\mathbf{p}_c$. Enforcing Equation (6) is straightforward with a position

based simulator [Müller et al. 2007] and with a force-based simulator can be achieved using constraint forces [Bergou et al. 2007]. Either way, the quality of the constrained fine scale simulation depends on the chosen test functions $\mathbf{T}$. For example, if we choose delta functions as test functions (i.e., the matrix $\mathbf{T}$ is zero except for exactly one 1 in each row), the fine simulation will exhibit objectionable tugging artifacts [Bergou et al. 2007]. The test functions proposed by Bergou and colleagues lead to more natural tracking, but small temporal artifacts are noticeable in certain frames, see Figure 6 (left).

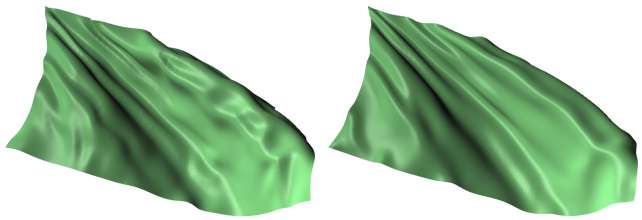


Figure 6: *Fine mesh tracked with 80 test functions obtained using clustering (left) compared to the same number of harmonic test functions (right).*

To analyze this issue, we observe the tracking constraint (6) requires the orthogonal projections of $\mathbf{p}_f$ and $\mathbf{B}\mathbf{p}_c$ on the row space of $\mathbf{T}$ to match. The component of $\mathbf{p}_f$ in the nullspace of $\mathbf{T}$ is not influenced by the tracking constraint. In other words, the corresponding degrees of freedom behave exactly as dictated by physics. Since we use tracking to restore fine-scale detail missing in the coarse simulation, we argue that the row space of $\mathbf{T}$ should contain only low frequencies, leaving the higher frequencies undisturbed. The assumption of orthonormal rows of $\mathbf{T}$ again leads us to low-order harmonics, already computed for harmonic regularization (see Figure 5). In our experience, harmonic test functions $\mathbf{T}$ lead to more natural tracking results because only the low frequencies are constrained and, therefore, interference with the high frequencies is avoided (see Figure 6 right). Additionally, the only parameter to tune is the number, T, of applied test functions, which has a very intuitive meaning—higher T leads to closer tracking. On the other hand, we note that spatially varying test functions [Bergou et al. 2007] offer greater flexibility and sparser tracking constraints. An interesting extension would be to consider how the generalizations of the Laplacian [Wardetzky et al. 2007] can yield anisotropic and/or spatially varying generalizations of harmonic functions.

5 Adding Oscillatory Modes

With training data generated using tracking (Section 4), the coarse vertex positions $\mathbf{p}_c$ are usually good predictors of the fine vertex positions $\mathbf{p}_f$. An exception occurs in situations with persistent external forcing, such as when environmental cloth (flags, sails, ...) is subjected to strong wind. In this case, even the tracked simulation often exhibits traveling waves that cannot be explained with $\mathbf{p}_c$. Below we propose a method to capture these effects by using *oscillatory modes* to add high frequency details after the application of the upsampling operator. Note that this approach is related to corrective techniques in skinning [Kry et al. 2002].

We start by examining the residual signal. For a given frame, we denote the residual vector $\mathbf{p}_r = \mathbf{p}_f - \mathbf{U}\mathbf{p}_c \in \mathbb{R}^{N\times 3}$. In simulations with persistent external forcing, we observe a strong traveling wave component. These traveling waves produce displacements mostly along the normal to the upsampled surface $\mathbf{U}\mathbf{p}_c$. Consequently, we project the residuals onto the surface normals. Letting $\mathbf{n}_i$ be the surface normal at vertex i we construct a displacement vector $\mathbf{d} \in \mathbb{R}^{N\times 1}$ such that $d_i = \mathbf{p}_{r,i}^T \mathbf{n}_i$ for $i = 1, \ldots, N$. We calculate the normal displacement vectors $\mathbf{d}$ for each frame and stack them in a matrix $\mathbf{D} \in \mathbb{R}^{N\times F}$. The same method could also be applied to tangential directions, if required.

Our goal is to decompose the matrix $\mathbf{D}$ into a spatial and temporal component, represented by matrices $\mathbf{E} \in \mathbb{R}^{N\times 2}$ and $\mathbf{\Omega}(\theta) \in \mathbb{R}^{2\times F}$, such that $\mathbf{D} \approx \mathbf{E}\mathbf{\Omega}(\theta)$. Because we want to model traveling phenomena, the temporal component $\mathbf{\Omega}(\theta)$ stores a pair of sine and cosine functions with the same frequency, i.e., its j-th column contains $(\sin(j\theta),\ \cos(j\theta))^T$ for $j = 1, \ldots, F$. The frequency $\theta \in \mathbb{R}$ depends on the speed of the traveling wave. The spatial component $\mathbf{E}$ can be interpreted as phase shifts and amplitudes at each vertex, because for every $e_{i,1}, e_{i,2} \in \mathbb{R}$ there exists $r_i, \phi_i \in \mathbb{R}$ such that $e_{i,1} = r_i \cos(\phi_i)$ and $e_{i,2} = r_i \sin(\phi_i)$. The product $\mathbf{E}\mathbf{\Omega}(\theta) \in \mathbb{R}^{N\times F}$ therefore reconstructs the traveling waves because for vertex i and frame j we obtain

$$r_i \cos(\phi_i) \sin(j\theta) + r_i \sin(\phi_i) \cos(j\theta) = r_i \sin(\phi_i + j\theta) \quad (7)$$

It remains to find the optimal $\mathbf{E}$ and θ for a given $\mathbf{D}$, i.e., solve

$$\underset{\mathbf{E},\theta}{\operatorname{argmin}} \|\mathbf{D} - \mathbf{E}\mathbf{\Omega}(\theta)\|$$

First, we find a suitable θ by performing autocorrelation on the columns of $\mathbf{D}$, discovering the strongest periodic component of the signal. Second, once θ has been determined, $\mathbf{E}$ can be calculated using linear regression. We also experimented with regularization but found overfitting is not an issue because high frequencies are desirable in this case. The whole process can then be repeated on the deflated version of $\mathbf{D}$, i.e., $\mathbf{D} - \mathbf{E}\mathbf{\Omega}(\theta)$, until its Frobenius norm decreases below a given threshold. However, in our flag example we found one pair of oscillatory modes to be sufficient, see Figure 7 visualizing our $\mathbf{E}$. Note that this process is closely related to Complex PCA [Kass and Anderson 2008], however, we do not use any complex arithmetics.

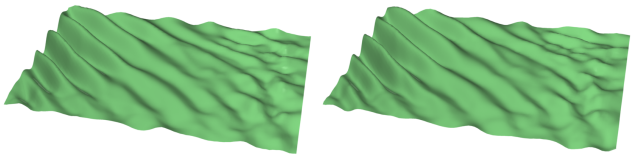


Figure 7: *A pair of oscillatory modes for the flag example describing phase shifts and amplitudes at every vertex.*

At run-time, the oscillatory modes are used to enhance the upsampled mesh $\mathbf{U}\mathbf{p}_c$. The matrix $\mathbf{E}$ is constant and the current column of $\mathbf{\Omega}(\theta)$ is given by the simulation time. The displacements reconstructed according to Equation (7) are multiplied by the normal $\mathbf{n}_i$ and added to vertex positions. In spite of their simplicity, the oscillatory modes provide interesting high frequency detail, see Figure 8.

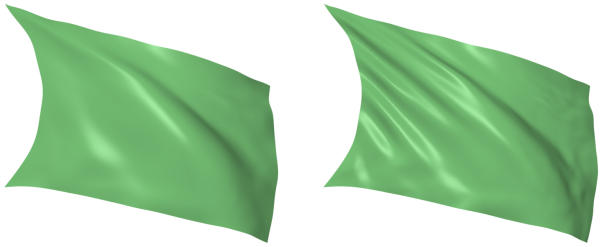


Figure 8: *Flag flapping in the wind upsampled without oscillatory modes (left) and with one pair of oscillatory modes (right).*

6 Implementation and Results

Simulation. Our approach is not tied to a specific simulator, but to generate our results we used our own implementation of position based dynamics [Müller et al. 2007]. To ensure gradual transition between kinematically controlled attachments, such as the waist edge-loop of the skirt, we smoothly transition between skinning and the upsampled mesh, similarly to Stoll and colleagues [2010]. We currently perform only simple collision detection using sphere-swept bounding volumes attached to the bones of an actor, as is often done in games. For good tracking results, we allow compression in the coarse-scale simulation, but not in the fine-scale one [Müller and Chentanez 2010; Rohmer et al. 2010].

Experiments. We designed four test scenarios focusing on free flowing and environmental cloth; clothing that closely follows the motion of the body can be handled with methods using the character rig [Kim and Vendrovsky 2008; Wang et al. 2010a; de Aguiar et al. 2010]. Each of our examples features one training and two novel testing (generalization) motions, see Table 1 and Figure 11. For the training sequence, an artist sets up the coarse and fine simulation parameters, the number of test functions and, after the simulation is complete, the regularization parameters. In our experience, the simulation parameters are typically the hardest to tune. The resulting upsampling operator is then tested on the two novel generalization motions (e.g., dance and jumping jacks) without generating any new training data. While automatic cross-validation techniques would also be possible, having an artist in the loop is usually preferred in game development. As shown in Figure 11, the upsampling operators result in higher visual quality than subdivision while requiring only a fraction of the time compared to the fine-scale simulation (computed on generalization motions for comparison only).

	Skirt	Flag	Curtain	Cape
Model statistics:				
M (coarse points)	196	150	121	98
N (fine points)	7016	6336	5041	10162
F (train frames)	1073	1386	1490	905
T (test functions)	60	80	80	50
Run-time:				
Memory	5.5MB	3.9MB	2.4MB	4MB
Upsampling on CPU	0.8ms	0.6ms	0.4ms	0.7ms
Upsampling on GPU	0.1ms	0.08ms	0.06ms	0.07ms
Coarse simulation	0.5ms	0.4ms	0.3ms	0.2ms
Pre-processing:				
Compute harmonics	111s	84s	41s	343s
Learn $\bar{\mathbf{U}}$ (our method)	6.8s	5.1s	3.4s	7.7s
Learn $\mathbf{U}$ (PARDISO)	N/A*	6410s	1263s	3637s

Table 1: *Model statistics, memory consumption, run-time performance and pre-processing times for our examples. *In the skirt example PARDISO ran out of memory (24GB).*

Run-time performance. The CPU simulation and upsampling times reported in Table 1 are measured on a 2.9GHz Intel X5670 CPU using a single thread. For the numerical computations used in pre-processing (last three rows in Table 1), we employ the multi-threaded Intel™MKL library. Additionally, we tried simulating the fine-scale meshes directly in Bullet, an optimized physics library frequently used in both the film and game industries. Its single thread CPU run-time performance was around 30ms per frame on the skirt mesh, which is far too slow for games. (The speed of our prototype simulator is comparable to Bullet.) However, coarse simulation is much faster and combined with our upsampling takes about 1ms on a single CPU core, which is sufficiently fast for games. We can also stream the coarse points $\mathbf{p}_c$ to the GPU (we used an NVIDIA GTX 280) and execute the upsampling there. The upsampling time on the GPU is very small and should be acceptable even in games that already have a GPU-heavy run-time component. The *stable spaces* approach [de Aguiar et al. 2010] offers performance comparable to our method; for example, they report a dress model with 5178 vertices running in 3.7ms on a CPU. However, our method generalizes better due to the run-time physics component, handles free flowing and environmental cloth and does not need to learn dynamics effects from the training data. Also, our training phase is faster and only requires a single training motion sequence (de Aguiar and colleagues [2010] use 33 motion clips and their learning phase takes 1.5 hours).

Regularization parameters. To define our polynomial profile for harmonic regularization (Section 3.2), we need a non-decreasing sequence $\gamma_1, \ldots, \gamma_N$ specified by the parameters $a, b, c \geq 0$. In our experiments, we first set the bounds γ_1 and γ_N, representing the minimal and maximal penalties (we typically use $\gamma_1 = 1$ and $\gamma_N = 10^5$). The coefficients a and b can be directly computed from γ_1, γ_N and the exponent c. The exponent c controls the shape of the resulting curve, with larger c allowing finer details in the upsampling operator, see Figure 9. However, finer wrinkles do not generalize well to novel motions, and therefore a suitable compromise must be found (we typically use $c = 4$).

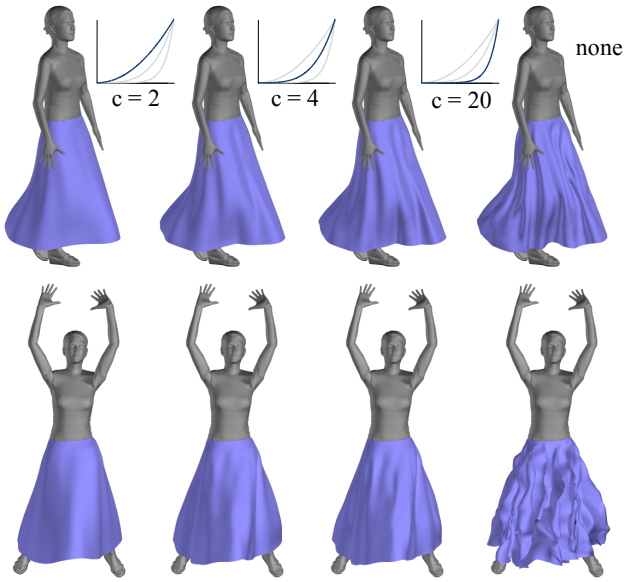


Figure 9: *The effect of varying the regularization parameter c in (top) the training walk sequence and (bottom) generalization motion. Compare with (right) no regularization.*

Generalization. With appropriately chosen regularization parameters, our upsampling operators can be applied to motions quite different from the training data. This feature is important, because in modern games the range of possible motions is often huge and constantly evolves during development. However, the ability to generalize has its limitations, as can be seen in Figure 10 (left) where an upsampling operator trained on a walking motion is applied to a dance motion—the diagonal wrinkles from torsional twists are not captured, because they did not appear in the training data. This limitation can be addressed by constructing a new operator $\mathbf{U}$ for the dance animation, see Figure 10 (right). An interesting extension would be to learn multiple versions of $\mathbf{U}$ and blend between them at run-time; however, all of our examples (see Figure 11) use only a single upsampling operator. We observed the oscillatory modes (Section 5) used to enhance the flag animation also generalize surprisingly well to different wind conditions and movements of the pole, even though, for extreme wind conditions the direction of the wrinkles is likely to be incorrect.

Figure 10: *Upsampling learned from walking and applied to dancing (left), compared to fine-scale tracking simulation (middle) and the result of a re-trained upsampling operator (using the same regularization parameters, right).*

Discussion. While the advanced nonlinear deformers [Feng et al. 2010; Wang et al. 2010a; Rohmer et al. 2010] are able to deliver a very high level of detail and finer features than our method, their performance is currently insufficient for games. The fastest of the three [Feng et al. 2010] runs for $M = 100, N = 19451$ in 3.7ms on the GPU; our method in the same settings requires 0.33ms (0.13ms upsampling + 0.2ms simulation) and is substantially easier to implement. At the other end of the spectrum, subdivision is faster than our technique and can generate smooth meshes of arbitrarily high resolution in any configuration. However, the isotropic nature of subdivision shape functions (see Figure 3a) limits the visual quality especially when using very coarse simulation meshes, resulting in non textile-looking deformations and lack of detail.

7 Limitations and Future Work

The upsampling operator may introduce penetrations even if the coarse simulation is collision free; we prevent such collisions by expanding the collision volumes. Our simulator also does not handle cloth self-collisions due to performance considerations; the low execution time of our method is achieved by using only a simple coarse simulation model and linear upsampling operators. In its current form, our approach is not suitable for much higher resolutions because the dense upsampling operators become prohibitively expensive. An interesting area of future work would be to further compress and/or sparsify the upsampling operators, e.g., explicitly enforce compact support of the shape functions. If finer-scale features are desired, the results of our upsampling operators can be further refined by procedural wrinkle synthesis [Rohmer et al. 2010; Müller and Chentanez 2010]. For example, Müller and Chentanez [2010] demonstrate refinement of simulation meshes ranging from 2k-7k vertices to very detailed meshes with 32k-112k vertices. Another limitation is that our upsampling operator uses only the current state of the coarse simulation and therefore does not capture any fine scale dynamics. An interesting extension would be to add previous states of the coarse mesh (history) to the regression, similarly to de Aguiar and colleagues [2010].

Our harmonic regularization is quite general and could be useful for domains other than cloth, for example, hand, facial and hair animation. In the context of cloth, we envision pre-visualization applications in feature animation, especially if the cloth design cycle already contains tracking—the artist could preview the coarse simulation upsampled using our operator (perhaps learned from a previous iteration) before committing resources to fine-scale tracking simulation. Overall, we believe our approach helps to fill the gap between extremely fast but non-physical subdivision operators and more complex non-linear deformers.

8 Acknowledgements

We thank all the reviewers for their feedback and helpful comments. Many thanks to our colleagues Olga Sorkine, Alexander Hornung, Bernd Bickel and Peter Shirley for careful proofreading and Daniel Sýkora, Robert Sumner, Edilson de Aguiar and Leonid Sigal for stimulating discussions. The human models were created by Peter Lozsek and were generously provided by Trinity College Dublin. We also thank Paul Johnson, Jeff Bunker, Brian Christensen and Yong Wan for help with modeling and art feedback and Rasmus Tamstorf, James O'Brien, Huamin Wang, Wei-Wen Feng and Pascal Volino for sharing their expertise on cloth simulation.

References

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 1998*, 43–54.

BARBIČ, J., AND JAMES, D. L. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph. 24*, 3, 982–990.

BARBIČ, J., AND POPOVIĆ, J. 2008. Real-time control of physically based simulations using gentle forces. *ACM Trans. Graph. 27*, 5, 163:1–163:10.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph. 28*, 3, 53:1–53:9.

BERGOU, M., MATHUR, S., WARDETZKY, M., AND GRINSPUN, E. 2007. TRACKS: Toward directable thin shells. *ACM Trans. Graph. 26*, 3, 50:1–50:10.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3, 594–603.

CORDIER, F., AND MAGNENAT-THALMANN, N. 2005. A data-driven approach for real-time clothes simulation. *Comput. Graph. Forum 24*, 2, 173–183.

CUTLER, L. D., GERSHBEIN, R., WANG, X. C., CURTIS, C., MAIGRET, E., PRASSO, L., AND FARSON, P. 2005. An art-directed wrinkle system for CG character clothing. In *Proceedings of the 2005 Symposium on Computer animation*, 117–125.

DE AGUIAR, E., SIGAL, L., TREUILLE, A., AND HODGINS, J. K. 2010. Stable spaces for real-time clothing. *ACM Trans. Graph. 29*, 4, 106:1–106:9.

DEROSE, T., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *Proceedings of SIGGRAPH 1998*, 85–94.

ENGLISH, E., AND BRIDSON, R. 2008. Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. 27*, 3, 66:1–66:5.

FENG, W.-W., KIM, B.-U., AND YU, Y. 2008. Real-time data-driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. 27*, 3, 91:1–91:9.

FENG, W.-W., YU, Y., AND KIM, B.-U. 2010. A deformation transformer for real-time cloth animation. *ACM Trans. Graph. 29*, 4, 108:1–108:9.

GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. 2007. Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3, 49:1–49:8.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *Proceedings of the 2003 Symposium on Computer animation*, 62–67.

HADAP, S., BANGERTER, E., VOLINO, P., AND MAGNENAT-THALMANN, N. 1999. Animating wrinkles on clothes. In *Proceedings of the 10th IEEE Visualization Conference*, 175–182.

HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. 2009. Asynchronous contact mechanics. *ACM Trans. Graph. 28*, 3, 87:1–87:12.

HERMAN, D. L. 2001. Using precomputed cloth simulations for interactive applications. In *SIGGRAPH 2001 Sketches*.

JAKOBSEN, T. 2001. Advanced character physics. In *Game Developers Conference 2001*.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. 22*, 3, 879–887.

JAMES, D. L., AND PAI, D. K. 2002. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph. 21*, 3, 582–585.

KANG, M. K., AND LEE, J. 2007. A real-time cloth draping simulation algorithm using conjugate harmonic functions. *Comput. Graph. 31*, 2, 271–279.

KANG, Y.-M., CHOI, J.-H., CHO, H.-G., AND LEE, D.-H. 2001. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer 17*, 147–157.

KASS, M., AND ANDERSON, J. 2008. Animating oscillatory motion with overlap: wiggly splines. *ACM Trans. Graph. 27*, 3, 28:1–28:8.

KAVAN, L., SLOAN, P.-P., AND O'SULLIVAN, C. 2010. Fast and efficient skinning of animated meshes. *Comput. Graph. Forum 29*, 2, 327–336.

KIM, T.-Y., AND VENDROVSKY, E. 2008. Drivenshape: a data-driven approach for shape deformation. In *Proceedings of the 2008 Symposium on Computer animation*, 49–55.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. EigenSkin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 Symposium on Computer animation*, 153–159.

LARBOULETTE, C., AND CANI, M.-P. 2004. Real-time dynamic wrinkles. In *Proceedings of Computer Graphics International 2004*, 522–525.

LAWSON, C. L., AND HANSON, R. J. 1974. *Solving Least Squares Problems*. Prentice Hall, Englewood Cliffs, NJ.

LÉVY, B., AND ZHANG, R. H. 2010. Spectral geometry processing. In *SIGGRAPH 2010 Course Notes*.

LOOP, C. 1987. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, University of Utah.

LOVISCACH, J. 2006. Wrinkling coarse meshes on the GPU. *Comput. Graph. Forum 25*, 3, 467–476.

MÜLLER, M., AND CHENTANEZ, N. 2010. Wrinkle meshes. In *Proceedings of the 2010 Symposium on Computer animation*, 85–92.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2, 109–118.

MÜLLER, M., JAMES, D., STAM, J., AND THÜREY, N. 2008. Real-time physics. In *SIGGRAPH 2008 Course Notes*.

MÜLLER, M. 2008. Hierarchical position based dynamics. In *Proceedings of the 5th Workshop on Virtual Reality Interactions and Physical Simulations*.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. *Comput. Graph. Forum 25*, 4, 809–836.

PIGHIN, F., AND LEWIS, J. P. 2007. Practical least-squares for computer graphics. In *SIGGRAPH 2007 Course Notes*.

PINKALL, U., AND POLTHIER, K. 1993. Computing discrete minimal surfaces and their conjugates. *Experiment. Math. 2*, 15–36.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface 1995*, 147–154.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph. 29*, 5, 157:1–157:8.

SCHENK, O., AND GARTNER, K. 2006. On fast factorization pivoting methods for symmetric indefinite systems. *Elec. Trans. Numer. Anal. 23*, 58–179.

SORKINE, O., COHEN-OR, D., IRONY, D., AND TOLEDO, S. 2005. Geometry-aware bases for shape approximation. *IEEE Trans. Vis. Comput. Graph. 11*, 2, 171–180.

STAM, J. 2009. Nucleus: towards a unified dynamics solver for computer graphics. In *IEEE Int. Conf. on CAD and Comput. Graph.*, 1–11.

STOLL, C., GALL, J., DE AGUIAR, E., THRUN, S., AND THEOBALT, C. 2010. Video-based reconstruction of animatable human characters. *ACM Trans. Graph. 29*, 6, 139:1–139:10.

THOMASZEWSKI, B., PABST, S., AND STRASSER, W. 2009. Continuum-based strain limiting. *Comput. Graph. Forum 28*, 2, 569–576.

TSIKNIS, K. D. 2006. *Better Cloth Through Unbiased Strain Limiting and Physics-Aware Subdivision*. Master's thesis, University of British Columbia.

VALLET, B., AND LÉVY, B. 2008. Spectral geometry processing with manifold harmonics. *Comput. Graph. Forum 27*, 2, 251–260.

VOLODINE, T., VANDERSTRAETEN, D., AND ROOSE, D. 2006. Smoothing of meshes and point clouds using weighted geometry-aware bases. *Lecture Notes in Computer Science 4077/2006*, 687–693.

WANG, H., HECHT, F., RAMAMOORTHI, R., AND O'BRIEN, J. 2010. Example-based wrinkle synthesis for clothing animation. *ACM Trans. Graph. 29*, 4, 107:1–107:8.

WANG, H., O'BRIEN, J., AND RAMAMOORTHI, R. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph. 29*, 6, 156:1–156:10.

WARDETZKY, M., BERGOU, M., HARMON, D., ZORIN, D., AND GRINSPUN, E. 2007. Discrete quadratic curvature energies. *Comput. Aided Geom. Des. 24*, 8-9, 499–518.

ZORIN, D., SCHRÖDER, P., DEROSE, A., KOBBELT, L., LEVIN, A., AND SWELDENS, W. 2000. Subdivision for modeling and animation. In *SIGGRAPH 2000 Course Notes*.

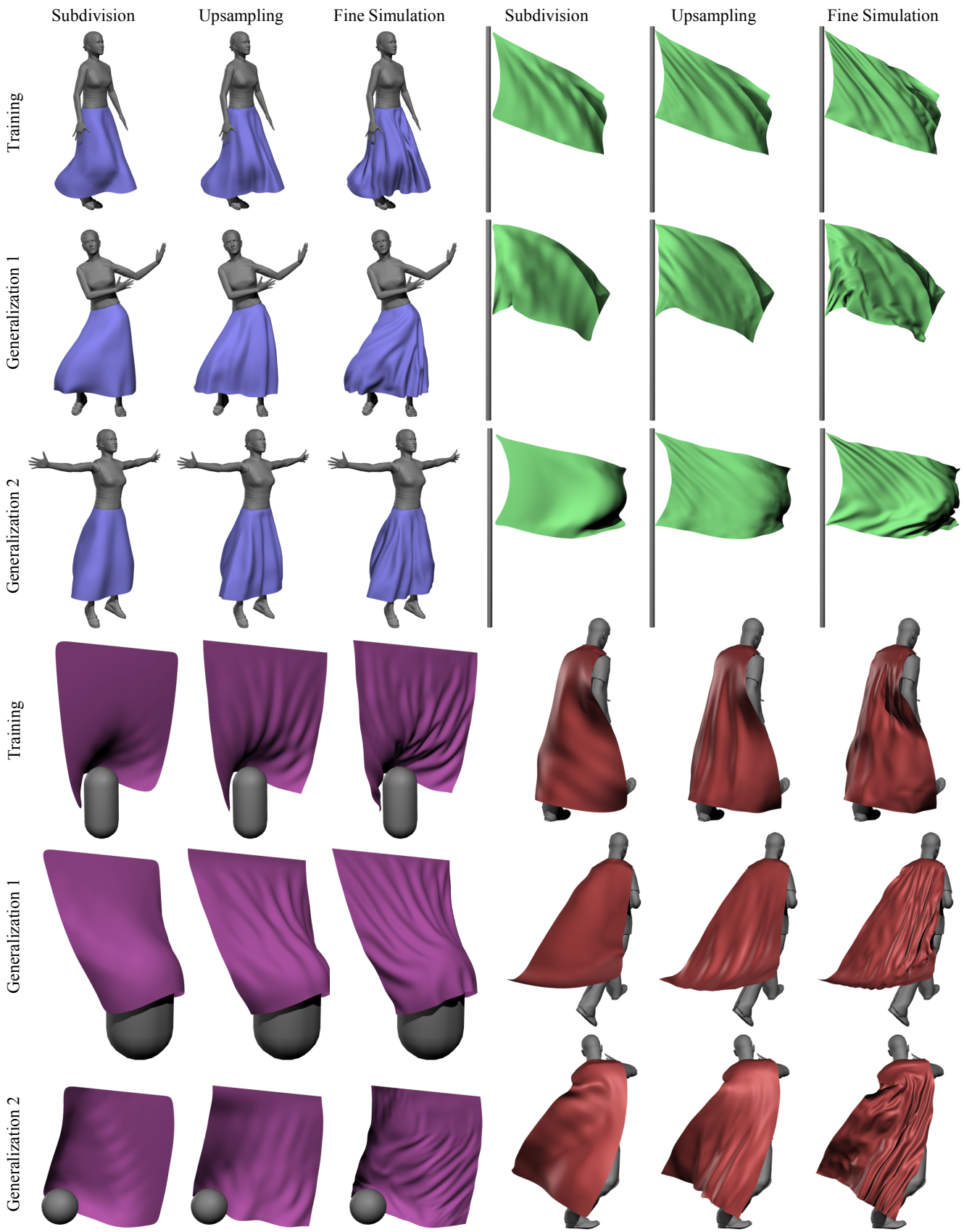


Figure 11: *Our upsampling method compared to subdivision and fine-scale physics in one training and two novel generalization motions: skirt (walk, dance, jumping jacks), flag (static, raise pole, fast wind), curtain (small object, big object, shooting balls), cape (walk, run, box). The high-resolution simulation was computed on novel motions only for comparison and was not used in training.*