Motion-driven Concatenative Synthesis of Cloth Sounds

Steven S. An Doug L. James Steve Marschner
Cornell University

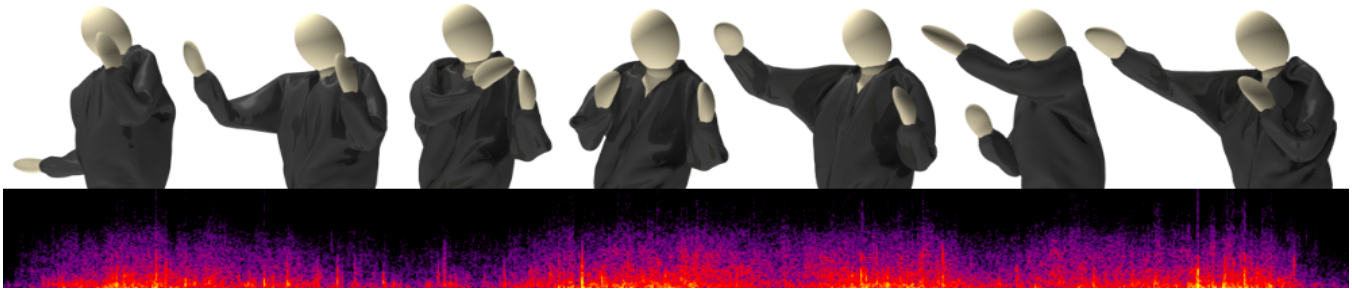


Figure 1: *Our data-driven approach to synthesizing cloth sounds is able to produce soundtracks for a wide range of common cloth animation scenarios. In this example, the familiar sounds of a windbreaker are synthesized as the character shadow boxes.*

Abstract

We present a practical data-driven method for automatically synthesizing plausible soundtracks for physics-based cloth animations running at graphics rates. Given a cloth animation, we analyze the deformations and use motion events to drive crumpling and friction sound models estimated from cloth measurements. We synthesize a low-quality sound signal, which is then used as a target signal for a concatenative sound synthesis (CSS) process. CSS selects a sequence of microsound units, very short segments, from a database of recorded cloth sounds, which best match the synthesized target sound in a low-dimensional feature-space after applying a hand-tuned warping function. The selected microsound units are concatenated together to produce the final cloth sound with minimal filtering. Our approach avoids expensive physics-based synthesis of cloth sound, instead relying on cloth recordings and our motion-driven CSS approach for realism. We demonstrate its effectiveness on a variety of cloth animations involving various materials and character motions, including first-person virtual clothing with binaural sound.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; H.5.5 [Information Systems]: Information Interfaces and Presentation—Sound and Music Computing;

Keywords: Sound synthesis, cloth simulation, data-driven methods

Links: DL PDF

1 Introduction

From the soft rustling of denim blue jeans or a woven cotton shirt, to the loud crumpling of a nylon windbreaker or the characteristic "zip" of corduroy pants, the natural sounds of clothing help bring virtual characters to life. Advances in computer graphics have enabled realistic visual simulation of cloth in computer animation and interactive virtual environments, but we still do not know how to automatically synthesize realistic sounds for synchronized accompaniment of these inherently silent cloth simulations.

Cloth animations pose unique challenges for digital sound synthesis. Cloth sounds are noisy, yet have a very natural and organic quality which is distinctively non-digital. People are also very familiar with clothing sounds, and so they can be quite attuned to the presence of digital synthesis artifacts. Direct numerical simulation of physics-based acoustic emissions from fabric is also highly complex, and would lead to expensive simulation times far beyond traditional cloth simulation. The need to simulate many different cloth materials, each with distinctive mechanical and sound properties, also complicates the estimation and tuning of parameters.

In this paper, we propose a *two-stage approach* for synthesizing cloth sounds that combines the responsiveness of motion-driven sound synthesis with the quality of real cloth recordings (see Figures 1 and 2). First, we devise a parametric cloth sound model driven by cloth motion that produces an initial *target sound* that captures important cues. Second, we use *concatenative sound synthesis* to piece together microsound *units*, from a database of pre-recorded cloth sounds, that best match the target sound.

Our parametric cloth sound model can be driven by input from graphics-rate cloth animations. The model accounts for two primary sources of acoustic emissions from cloth: frictional contact sounds and crumpling sounds. The friction noise and crumpling sound models are built using data from experiments in which we record the sound of a specimen of a specific material as it undergoes specific motions. We then analyze the input cloth animation's motions, extracting sliding contact and curvature information, and use spectral and sample-based synthesis techniques to generate sound with characteristics matching those observed in the experimental data. The model produces a synchronized sound that mimics variations in the true sound, such as synchronized crumpling events, and friction noise dependence on sliding speed. However, due to model limitations, its realism is limited beyond specific controlled animations. Therefore, we only use this model to generate a low-

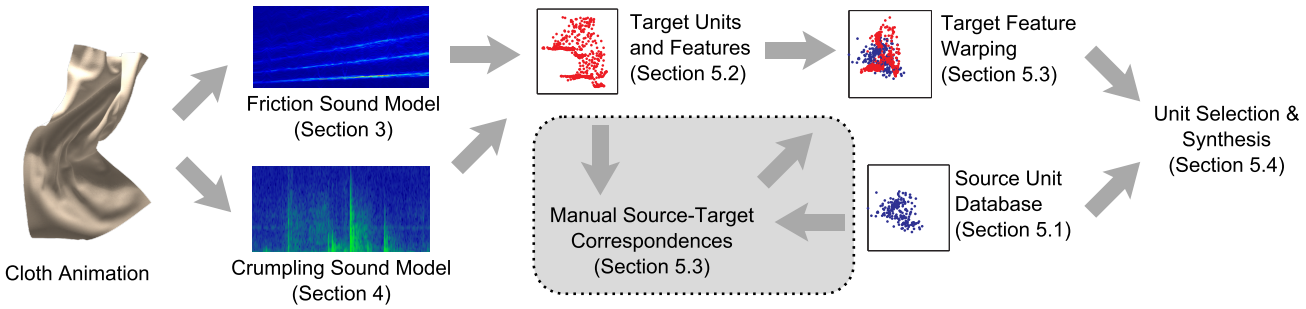


Figure 2: Overview: *Given a cloth animation, we first use two parametric sound models for friction and crumpling sounds to synthesize a low-quality "target" signal. We then dice the signal up into short sound "units" and compute per-unit feature vectors. Finally, we warp the features using a manually tuned warping function (which can be reused) and use a unit selection process to select a nearby high-quality "source unit" from a database to replace every target unit. The selected units are concatenated to synthesize the final cloth sound.*

quality *target signal*, which in turn drives the second synthesis process based on *concatenative sound synthesis* (CSS) [Schwarz 2004]. CSS is used to piece together microsound *units* from a database of pre-recorded relevant sounds. We select a sequence of database units which best matches the sequence of target sound units. This *unit selection* process determines the distance between target and source units by using low-dimensional feature vectors based on mel-frequency cepstrum coefficients (MFCC). Since our synthesized target signal can differ significantly from the recorded database sounds, we warp the target's feature vector space with a manually tuned warping function to better match the variations in the database. The final synthesized cloth sound is then generated by concatenating the selected sound units together, with minimal filtering to avoid introducing digital artifacts.

Our data-driven method can produce plausible cloth sounds, and we can render realistic first-person cloth experiences by using low-noise binaural microphones for database recording. Our examples demonstrate its effectiveness with the challenging examples of corduroy pants and nylon windbreakers, as well as non-character examples such as blankets and sheets (cotton and polyester). The CSS model is built using a specific material and garment undergoing particular actor motions along with a calibration dataset of simulated cloth motions; however, the model can be reused for novel simulated cloth motions at runtime.

Related Work: Cloth simulation is widespread in computer animation, and a variety of cloth models have been proposed and studied with the aim of visual reproduction of cloth behaviors [Terzopoulos et al. 1987; Courshesnes et al. 1995; Baraff and Witkin 1998; Kaldor et al. 2008]. In addition to the underlying models, many methods and algorithms for integrating their dynamics also exist with varying trade-offs between efficiency and accuracy [Baraff and Witkin 1998; Choi and Ko 2002], and strategies for resolving collisions and contact [Bridson et al. 2002]. Unfortunately, such visual simulation methods are not inherently well-suited to resolving the acoustic vibrations of cloth.

Recently there has been increased interest in developing sound synthesis techniques for computer animations and virtual environments. Techniques vary in how much they are based on physical principles as opposed to recorded data. Many early synthesis techniques are based on simplified models of musical instruments, such as guitar strings and vibrating membranes [Karplus and Strong 1983; Bilbao 2009]. Recently, much work has been done on synchronizing physically based synthesis techniques with physically based animations. Rigid bodies are well-approximated by efficient linear modal synthesis techniques [van den Doel and Pai 1996; van den Doel et al. 2001; O'Brien et al. 2002; James et al. 2006; Raghuvanshi and Lin 2006; Zheng and James 2010], which unfortunately provide poor approximations to cloth sounds. Some types of clothing, for example plastic windbreakers, are well modeled as thin shells. Nonlinear thin-shell and plate sounds have been widely considered (see [Bilbao 2009; Chadwick et al. 2009]), but such sound models are typically only valid for very small deformations and cannot support crumpling. O'Brien et al. [2001] proposed a method for synthesizing sounds for general FEM-based simulations, and sheet-like examples were considered. However, due to high computational costs and difficult parameter tuning, we seek a method that is more practical and easier to control. It is also unclear if existing cloth models could model the mechanics of crumpling and friction between yarns well enough to produce plausible sound vibrations.

On the other end of the spectrum, data-driven methods focus on using, or reproducing characteristics of, recorded data when the underlying physical systems are either too expensive to simulate or methods simply cannot produce convincing sound [Cook 2002; Picard et al. 2009; Peltola et al. 2007]. These techniques often utilize general synthesis algorithms, such as inverse-FFT synthesis [Rodet et al. 1992; Marelli et al. 2010], and additive and subtractive synthesis [Serra and Smith 1990]. Our technique uses several of these techniques to synthesize low-quality target sounds based on sliding noise and crumpling events.

Sound texture modeling and synthesis methods can be used to resynthesize an audio corpus [Dubnov et al. 2002; Strobl et al. 2006]. Chadwick and James [2011] synthesized fire sounds using a hybrid sound synthesis approach wherein a low-frequency sound is first generated from a physics-based simulation, then data-driven sound texture synthesis is used to add high-frequency details. In contrast, we use cloth motion to drive a low-quality measurement-based sound model, then use that to generate a target signal for use with concatenative sound synthesis. Our method shares some similarities with the "Sound-by-Numbers" algorithm, [Cardle et al. 2003] which drives sound models using low-dimensional motion signals, such as the 2D position of a moving object. In contrast, 3D cloth animations produce high-dimensional motion signals.

Inspired by granular synthesis techniques [Roads 2004], our final sound is a concatenation of microsound units from cloth sound recordings. However, the particular selection of units is controlled by our low-quality synthesis model, and is most closely related to concatenative sound synthesis (CSS) [Schwarz 2004]. CSS has origins in speech sound synthesis [Hunt and Black 1996], where sound quality requirements dictates a data-driven approach. CSS has been used in computer music applications for musical instrument synthesis and for resynthesis of audio [Schwarz 2004]. In such cases, the

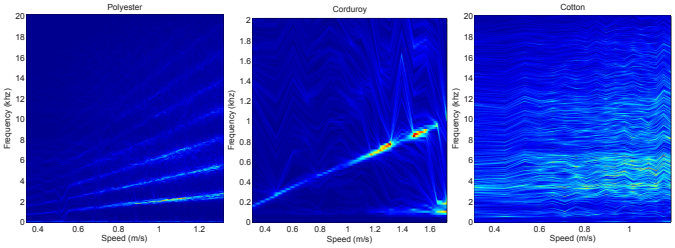
target signal can be symbolically represented, e.g., as a a sequence of phonemes or musical notes, as well as using target audio, such as in audio resynthesis.

The particular topic of synthesizing cloth sounds has seen very little work. Cho et al. [2001; 2005] performed experimental studies concerning the characteristics of frictional sounds and how they affect the perceived qualities of fabrics. Huang et al. [2003] synthesized sounds for a stylus being rubbed over a cloth patch using a modal model driven by measured roughness profiles. While appropriate for their particular haptic application, the model is unsuitable for general cloth animations. Crumpling sounds have received attention in the physics community due in large part to the interesting self-organized critical phenomena exhibited by acoustic emissions from crumpling events and their characteristic power-law statistics [Houle and Sethna 1996]. Such models have inspired geometry-independent stochastic sound models of crumpling [Fontana and Bresin 2003].

In motion pictures and video games, cloth sounds are often generated using hand-selected recordings, or "acted out" by foley artists. Such approaches can produce high-quality results, but are not computer automated, e.g., for interactive virtual environments. When they are automated, such as when triggered by computer-generated events [Takala and Hahn 1992], the sounds lack nontrivial dependence on the cloth's deformation and contact state. In a sense, our approach automates the foley process by converting digital cloth motion directly into a plausible sequence of cloth recordings.

2 Friction Sound Model

Sliding friction is an important component of cloth sound and is common when cloth rubs against itself or other surfaces. In this section, we describe a frequency-domain noise model that provides a material-specific approximation of friction sound. A primary behavior we wish to capture is how the frictional sound changes with respect to sliding speed, since this can introduce pitch-like variations in many materials (see Figure 3). While other factors can be important, such as geometric shape, contact state, tension, and scattering, it can be difficult to devise practical experiments to model these variations. As a first approximation to friction-driven sound, we use experiments to build a colored-noise model with parametric dependence on sliding speed. The model suffices to synthesize a semi-plausible noise-like sound for cloth in sliding contact for purposes of generating a target sound.

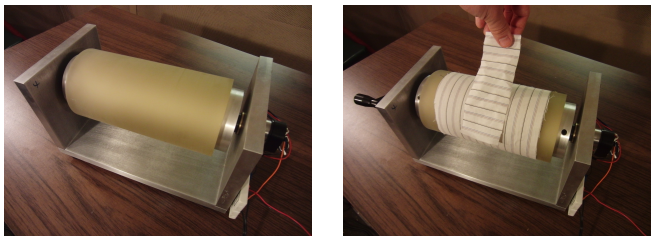


Polyester Corduroy Cotton

Figure 3: Friction sound spectra vs. sliding speed *are shown for several materials using the interpolated spectral sound model. Strong dependence on sliding speed is evident for some materials.*

Motion Analysis: Given the input cloth motion, our friction sound model is parameterized by sliding contact velocities. Sliding contact events can be generated by cloth simulators; however, for generality, and for use with commodity simulators where contact state is often inaccessible, we choose to estimate sliding contact events via position-based collision analysis of animation frames. For simplicity, we only consider point-triangle contacts. To accommodate different contact gap tolerances and interpenetrations, we assume that each point has some finite collision radius r specified by the user. If there are triangles within distance r of a vertex, we record a sliding contact with the closest triangle, and estimate relative contact speed using a forward-difference scheme. To interpolate and avoid harsh discontinuities, we fit piecewise cubic Hermite interpolating splines to the raw sliding velocities. The result of the analysis is a function $s_v(t)$ that returns the speed at which vertex v was sliding at time t. If it was not in contact with anything, we set $s_v(t) = 0$ (which will produce no sound). Similarly we filter out all points which never slide above some given speed threshold s_{thresh}, setting their speed to zero, which helps eliminate contacts merely due to proximity in the rest pose. We observe that higher mesh resolutions tend to produce cleaner sound signals due to improved spatiotemporal sampling of contact events.

Experimental Analysis: To obtain calibrated measurements of cloth friction sounds versus sliding speed, we constructed a specialized apparatus for frictional cloth sound measurement (see Figure 4). We wrap a rectangular piece of cloth around a mechanical roller, then spin the roller while manually holding another smaller piece of cloth in contact with it. The roller slows as friction dissipates its momentum, and the resulting sound is recorded until the roller comes to rest. Because cloth sounds can be very quiet at low sliding speeds, it is important to minimize the presence of other sounds when recording an experiment. All our recordings were done in a sound isolation room (*Industrial Acoustics Company controlled acoustical environment*) using equipment with low levels of self-noise: *RODE NT1-A* microphones and the *TASCAM DR-680* digital recorder. High-quality bearings help minimize the additional noise of the device itself. An optical encoder attached to the roller is used to accurately estimate the rotational speed at any given time. Since the cloth wrapped around the roller has a seam which can introduce sound artifacts, we use the encoder information to ignore sounds during seam contact. This setup effectively gives us a mapping from sliding speed to friction sound.



Device Usage

Figure 4: Apparatus for measuring cloth friction sounds *used to build a parametric friction noise model. A sample of cloth is wrapped around the roller, and another sample is held against it. The roller is spun and the friction sounds are recorded. An optical encoder on the roller provides synchronized sliding speed, $s(t)$.*

Parametric Noise Model: We begin by extracting a number of short clips (50 ms in all our examples) from the recorded sound convolved with a triangular window and taking the amplitude Fourier transform of each. We look up the speed in the encoder data at the center of each clip's time interval, resulting in a set of $(s_i, \mathcal{A}_i)$ pairs, where s_i is the sliding speed and $\mathcal{A}_i(f)$ is the amplitude of the Fourier transform coefficient for frequency bin f. These single-speed spectra are then interpolated to construct a material-specific estimate of $\mathcal{A}(f, s)$ for continuously varying s (see Figure 3).

Given a sliding speed s, we interpolate the two nearest amplitude spectra, $\mathcal{A}_i$ and $\mathcal{A}_j$, with interpolation parameter $\alpha(s) = (s - s_i)/(s_j - s_i)$. Direct linear interpolation using $\mathcal{A}(f, s) =$

$(1-\alpha)\mathcal{A}_i(f) + \alpha\mathcal{A}_j(f)$ produces an unsatisfactory effect of two different noise sources being blended together, which poorly captures pitch-changing "zipping" sounds. A better method is to interpret the spectra as probability distribution functions (PDFs), and linearly interpolate their *inverse cumulative distribution functions (CDFs)* instead [Matusik et al. 2005] (see Figure 5). Specifically, we approximate $\mathcal{A}(f,s)$ using

$$c_k(f) = \int_0^f \mathcal{A}_k(g)\,dg, \tag{1}$$

$$c_\alpha^{-1} = (1-\alpha)\,c_i^{-1} + \alpha\,c_j^{-1}, \tag{2}$$

$$\mathcal{A}(f,s) = \frac{dc_\alpha}{df}(f). \tag{3}$$

This method provides a smooth and perceptually pleasing way to interpolate between two amplitude spectra.

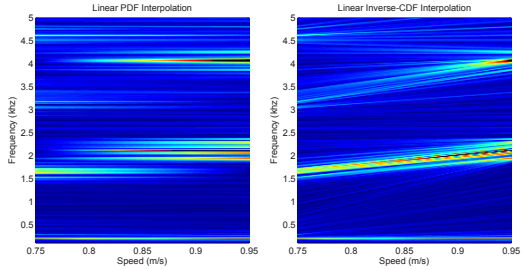


Figure 5: Interpolating Noise Spectra: *An example to illustrate the difference between interpolating the spectra as PDFs (Left) and interpolating the inverse CDF (Right). Interpolating PDFs corresponds to cross-fading two noise sources, whereas interpolating inverse CDFs yields a noise that shifts in pitch.*

Sound Synthesis: Using our parametric noise model, we can synthesize a soundtrack for each vertex v sliding with speed $s_v(t)$ at time t. We use the Inverse Fourier Transform (IFT) noise-synthesis method [Serra and Smith 1990; Rodet et al. 1992; Marelli et al. 2010] where the output signal is synthesized as a sequence of 50ms clips that overlap by 25ms. For a given clip centered at time t, we assume a constant amplitude spectrum $\mathcal{A}(f, s_v(t))$. We synthesize noise using this spectrum by using random phases to yield complex frequency-domain coefficients, then perform the IFT. We also use the vertex's position at time t to perform any position-based auralization; in our examples, we apply an HRTF model [Brown and Duda 1998]. Lastly, we overlap and add the clips. The final friction sound is the sum of all per-vertex friction sounds. While this method ignores many other factors that add variation to frictional sound, such as transfer effects and variations due to tension and contact forces, it suffices to produce an initial low-quality target sound.

3 Crumpling Sound Model

As well as making frictional sounds, cloth can also buckle and produce crumpling sounds, which can sound like small "pops." Woven garments, such as dress shirts and denim jeans, produce audible crumpling sounds, and stiff synthetics, such as nylon windbreakers, exhibit characteristically loud crumpling sounds. By analyzing curvature changes in the input cloth animation, we estimate buckling events and an energy-like measure, and use this information to drive a data-driven crumpling sound model for target sound synthesis.

Motion Analysis: Given an input cloth animation, which simulates crumpling phenomena to varying degrees of accuracy, we analyze it to estimate the time, location and size of crumpling events. We resort to a simple heuristic based on mean curvature to decide when a crumpling event has occurred and how much energy was involved in it. We consider a vertex v to be "buckling" at frame time t if its mean curvature H_v^t changed sign from frame $t-1$ to t. If it changes from negative to non-negative, call it a "positive buckle," and the opposite direction is a "negative buckle" (see Figure 6). Once every positive (negative) buckling vertex of frame t is identified, we take the graph consisting only of positive (negative) buckling vertices and edges incident to them. We find all connected components of such graphs and consider each component $\mathcal{C}$ to be a single event. Effectively, if a whole region of the cloth surface is buckling, this treats it as one large event rather than many small events. This produces a list of pop events $(t, \boldsymbol{p}, E)$, where $\boldsymbol{p}$ is the centroid of $\mathcal{C}$ and E is an energy-like measure of curvature changes

$$E = \left(\sum_{v\in\mathcal{C}} (H_v^t - H_v^{t-1})\right)^2 = \|H_{\mathcal{C}}^t - H_{\mathcal{C}}^{t-1}\|_1^2. \tag{4}$$

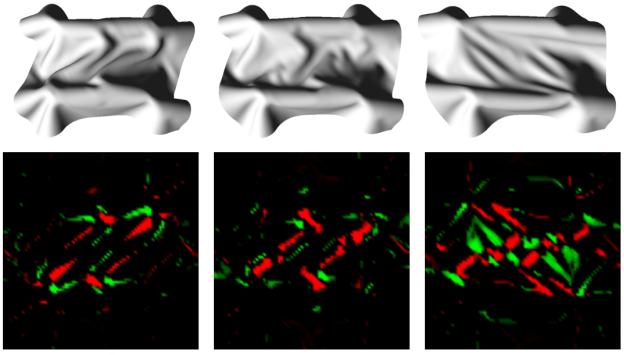


Figure 6: Crumpling Motion Analysis: *(Top) Three consecutive frames of a cloth simulation. (Bottom) Visualization of "buckling" vertices where black indicates no buckling, red a "negative buckle," and green a "positive buckle." Each contiguous region of red or green is treated as a single crumpling event.*

Experimental Analysis: We recorded crumpling sound events for use in data-driven sound synthesis. To isolate crumpling sounds, one must take care to minimize sliding contact sounds. We mount a $30cm$ square of the cloth on two metal handles (using magnets to hold it in place), then manually deform the cloth using an up-and-down shearing motion (see Figure 7). This setup consistently produced many crumpling sounds without friction sounds. It also provides a direct path to the microphone, unlike the cylindrical method of [Houle and Sethna 1996]. We typically record about 10-20 seconds of crumpling.

Figure 7: Crumpling Experiment: *To record isolated crumpling sounds of a given material, we affix a square swatch to metal handles using strong magnets, then manually shear the sample.*

During post-processing we extract recorded samples of individual crumpling events using the same method as [Houle and Sethna 1996]: we convolve the energy (the sum of squared pressure values) with a 20ms rectangular window and look for consecutive runs with response greater than a threshold E_{thresh}. We then extend each sample's extents to zero-crossings of the signal. See Figure 8 for an example of extracted samples. The E_{thresh} values used for

polyester, cotton, and the windbreaker were 0.25, 0.50, and 0.025, respectively. Buckling in corduroy was too quiet to merit modeling.

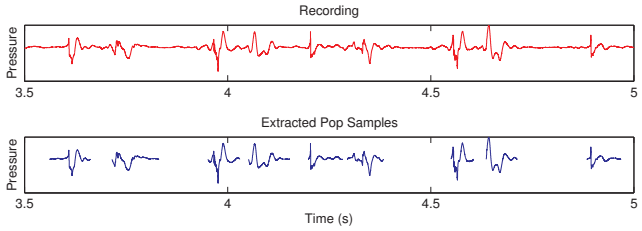


Figure 8: Crumpling sound events *extracted from a crumpling experiment recording (cotton) by an energy thresholding method.*

Data-driven Sound Synthesis: Given a crumpling event extracted from motion analysis, we assign a recorded crumpling sound with a similar "energy." We then simply play the sample when the event occurs to produce the output signal, with any subsequent auralization for stereo effects (we use an HRTF model [Brown and Duda 1998]). To avoid frame-rate-dependent artifacts, we also randomly jitter the playback time by 1/30 second. Since the "energy" values from motion events and sound samples are very different quantities, we use histogram matching [Heeger and Bergen 1995] to map an event energy E_e into a sample energy E_s, then use the sample with the nearest energy. Histogram matching essentially transforms the event distribution function $p_e(E_e)$ into the sample distribution $p_s(E_s)$ by evaluating $E_s = c_s^{-1}(c_e(E_e))$ where c_* is the cumulative distribution function (CDF) of p_*. We typically use 20 bins to compute the PDF, and approximate the CDF as a piecewise linear function. To reduce numerical noise artifacts and control the number of crumpling events, we reject all E_e events below a threshold E_{min}. Histogram matching can be done for each animation individually, but for online applications (where p_e is unknown *a priori*) we use p_e and E_{min} from a pre-existing "calibration animation" which contains representative crumpling motions.

4 Concatenative Synthesis of Cloth Sound

Given a cloth animation and the aforementioned friction and crumpling sound models, we can now synthesize a *target sound*,

$$X(t) = X_{friction}(t) + X_{crumpling}(t). \quad (5)$$

Unfortunately, while the target signal captures certain cloth characteristics and synchronized variations, it lacks the realism of real cloth recordings. Inspired by *Concatenative Sound Synthesis (CSS)* [Hunt and Black 1996; Schwarz 2004], we construct an improved result by concatenating samples from a source database of relevant cloth recordings so that they best match the target sound. We dice both the target signal X and the source signal U into short sound *units*, $(x_i), i = 1 \ldots n$ and (u_j), respectively, using short non-overlapping windows (referred to as an arbitrary grain segmentation). For all materials except corduroy, we found 4.2ms units to be long enough to maintain the characteristics of the recorded sounds. For corduroy, we used 16.7ms units to accommodate its larger-scale temporal structure. We hypothesize that there exists a target-to-source unit mapping $j = J(i)$, found using *unit selection*, such that the signal of concatenated source units, $S = (u_{J(1)}\ u_{J(2)}\ \cdots\ u_{J(n)})$, is a plausible soundtrack for the cloth animation. To facilitate comparison of target and source units for unit selection, we compute descriptive feature vectors $\mathbf{f}(x)$ for each sound unit x (or u). Since the target and source signals can be quite different, we nonlinearly warp the target feature vectors, $\mathcal{W}(\mathbf{f}(x))$, so that they better match the source database's feature vectors $\mathbf{f}(u)$ (§4.3). Finally, we perform unit selection to determine $J(i)$, essentially by minimizing the distance from $\mathcal{W}(\mathbf{f}(x_i))$ to $\mathbf{f}(u_{J(i)})$ (§4.4).

4.1 Database Acquisition

We construct databases using source recordings for specific materials, and particular cloth-character motion scenarios. For a richer database, we recorded each session using three microphones placed about a meter apart, and concatenated the signals into a single source signal. Images of the acquisition process are shown in Figure 9. Binaural recordings, which are ideal for self-sound listening experiences, were made using ultra-low-noise in-ear binaural microphones (*Sound Professionals, MS-TFB-2*). The exact motions recorded were chosen to cover calibration animations. For example, when recording the windbreaker, the subject wore a windbreaker and performed many jogging, flexing, punching, and waist-twisting motions. While a very large database can in principle improve the quality and range of the synthesized sound and avoid repetition, the target synthesis model is itself of limited fidelity so there are diminishing returns. In our examples, we typically recorded subject motions for about 1 minute.

Figure 9: Database acquisition: *(Left) Various natural motions and garments were recorded in a sound isolation room. (Middle) A cotton sheet is lightly waved back and forth. (Right) Punching motions while wearing a windbreaker.*

4.2 Feature Vectors

Low-dimensional feature vectors are used to summarize target and source sound units and facilitate easy comparisons between otherwise high-dimensional sound units. We found that mel-frequency cepstrum coefficients (MFCC) (see [Rabiner and Juang 1993]) provide effective descriptors for cloth sound units. The mel-frequency cepstrum is a popular compressed signal representation in speech applications. It works by taking the short-time Fourier transform, aggregating the spectral energy into uniform bins on the mel-frequency scale (an empirically derived scale meant to better match how humans distinguish pitch), and then taking a discrete cosine transform (DCT) of the logarithm of the bin energies:

$$\mathbf{f}(x) = \mathrm{DCT}(\log(\mathrm{mel}_N(x))), \quad (6)$$

where "mel" returns a spectral energy histogram with N bins spaced uniformly according to the mel scale.

This feature characterizes a unit's spectral shape and magnitude, and we find that $N = 3$ coefficients suffice to distinguish relevant characteristics of our sounds. Roughly speaking, the first coefficient correlates with loudness, and the other two coefficients correlate with how "crumply" a sound is (see Figure 10). It is unclear whether more coefficients would be helpful, and this is a topic of potential future work. For binaural recordings we concatenated the N-vector features of left and right channels to obtain a $2N$-vector feature.

4.3 Feature Warping

To facilitate unit selection, the feature-space distribution of target units must be warped to overlap well with that of the source units

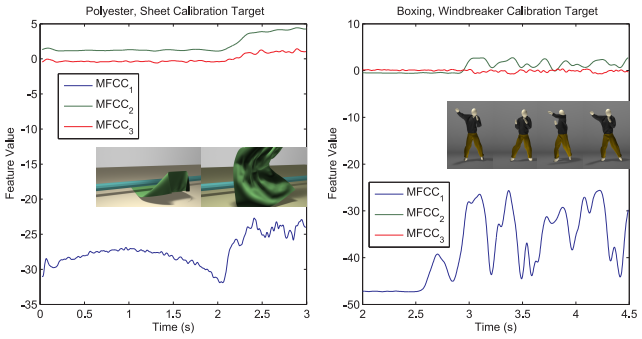


Figure 10: MFCC features vs. time: *Two plots illustrate how the 3 MFCC coefficients correlate with various motions. (Left) Unlike $MFCC_1$, $MFCC_2$ is unaffected by the variation in frictional sound for the first 2 seconds, but it fires when the shaking begins and crumpling sounds occur. (Right) We observe distinct peaks corresponding to punches and arm motion. Both $MFCC_1$ and $MFCC_2$ exhibit peaks for motions that are loud and crumply.*

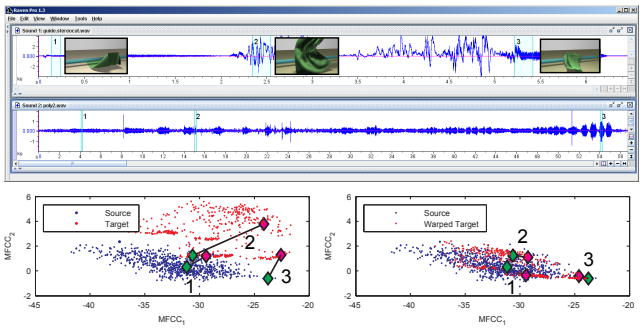


Figure 11: Manual selection of sound correspondences *is done using a simple interface (Top) with the target signal on top and the database source on the bottom. A sound designer selects target clips during perceptually important events, then selects database clips they would prefer to hear during such events. (Bottom) Plots of source and target units in feature-space (left/right showing before/after warping), along with larger markers at centroids of the selected clips. Target clips are magenta and source clips are green. Note that the correspondences are not exactly fit by the warping due to regularization.*

in the database. Given that these distributions can differ wildly depending on the target and recorded sounds, and that the "correct" target-to-database mapping is subjective and an opportunity to introduce stylization, we propose a user-guided approach. We synthesize a target signal for a set of training animations, then a sound designer provides a number of manual correspondences between parts of the target signal and the source/database signal. Given these correspondences, we fit a feature-warping function $\mathcal{W}(\mathbf{f}(x_i))$ based on thin-plate splines (see Figure 11) which can be reused for novel target signals. Filtering is used to make the process robust to noise and outliers. We now describe the process in detail.

Temporal Filtering: We temporally filter target and source features, since they can be noisy, e.g., due to crumpling events, and noisy features can lead to overfitting and bad feature matching for crumpling. We filter the features in time using a Gaussian filter; filter widths for each dimension are listed in Table 3.

Specifying Correspondences: Each correspondence is user-specified as time intervals (t_k, d_k, Δ_k), where for the k-th correspondence, t_k is the start of the interval in the target signal, d_k is the start in the database signal, and Δ_k is the length. For example, the most trivial correspondence is between intervals of silence in the target and database signals. Others might be between rapid sliding events and various degrees of desired crumpling. The intervals used in our examples are typically $\Delta = 100$ ms to $\Delta = 200$ ms long, and each fitting uses 5 to 15 correspondences. Figure 11 shows the interface we use for correspondence specification. Typically, it takes around 5 to 15 minutes of manual work to specify such correspondences, although it can take 2-5 iterations to achieve desirable results. See Section 6 for further discussion on this process. Given these correspondences (t_k, d_k, Δ_k), we find all units in these intervals and assume they correspond to each other in order. If $\{x_{i_1} \dots x_{i_n}\}$ is the set of target units which lie within the time interval $[t_k, t_k + \Delta_k]$ and $\{u_{j_1} \dots u_{j_n}\}$ is the set of database units in $[d_k, d_k + \Delta_k]$, then we assume the unit correspondences to be $C_k = \{(i_1, j_1) \dots (i_n, j_n)\}$. We collect these lists of unit-pairs C_k for all given correspondences and concatenate them into C.

Warp Function Fitting: Given the n correspondences $C = \{(i_1, j_1) \dots (i_n, j_n)\}$ between target and source units, we seek a smooth feature-warping function $\mathcal{W}$ that minimizes the error:

$$\sum_{k=1}^{n} \|\mathcal{W}(\mathbf{f}(x_{i_k})) - \mathbf{f}(u_{j_k})\|_2. \quad (7)$$

To warp the target unit features so that correspondences are respected, while also ensuring a smooth warp that avoids overfitting, we use a *regularized thin-plate spline* [Wahba 1990] similar to the approach taken by Belongie et al. [2002]. For a d-dimensional feature $\mathbf{f} = [\mathrm{f}_1 \dots \mathrm{f}_d]^T$, the displacement function for dimension l is

$$\Delta\mathbf{f}_l(\mathbf{f}) = A_{0,l} + \sum_{k=1}^{d} A_{k,l}\,\mathrm{f}_k + \sum_{k=1}^{n} W_{k,l}\,U(\|\mathbf{f} - \mathbf{f}(x_{i_k})\|_2) \quad (8)$$

where $U(r) = r^2 \log(r^2)$. The complete warping function is then

$$\mathcal{W}(\mathbf{f}) = \mathbf{f} + \Delta\mathbf{f}(\mathbf{f}). \quad (9)$$

The affine coefficients $A_{k,l}$, $k = 0 \dots d$, and nodal weights $W_{k,l}$, $k = 1 \dots n$, are obtained by solving a regularized linear system [Belongie et al. 2002; Wahba 1990]. We heavily regularize the system to avoid over-fitting and introducing unintended distortion; values for our regularization parameter λ (from equation 10 of [Belongie et al. 2002]) are given in Table 3.

Warp Reuse: In practice, we fit the feature warp once for a particular source database and calibration animation, then reuse it for novel target signals. This reuse reduces the need for manual intervention, and avoids example-specific tuning. Assuming the novel animations do not deviate too drastically from the feature-space covered by the calibration animation, the same warp will still produce plausible sounds. Thus artist intervention can be done just once per database–simulation pair to train the unit matching model, but an artist can still adjust warping to stylize the sound model.

4.4 Unit Selection & Synthesis

Next we select a database unit u_j for each target unit x_i by determining a selection function $j = J(i)$. Our final algorithm for unit selection is given below in Algorithm 1, which we now explain. Let the feature vector for the database unit u_j be $\mathbf{f}_j = \mathbf{f}(u_j)$, and let the warped and filtered target feature vector for unit x_i be $\tilde{\mathbf{f}}_i = \mathcal{W}(\mathbf{f}(x_i))$. The simplest unit selection method is to use the nearest neighbor given some distance metric:

$$J(i) = \arg\min_j D_1(i, j) = \arg\min_j \|\tilde{\mathbf{f}}_i - \mathbf{f}_j\|_2. \quad (10)$$

However, we find that it is often beneficial to use a contiguous sequence of units, $\{u_j, u_{j+1}, \dots, u_{j+L}\}$ from the database to preserve important temporal structure of the original recording. For

example, with corduroy pants, the zipping sound exhibits temporal structure larger than the unit size, but using larger units would restrict CSS flexibility. We use a simple greedy approach to encourage the selection of long sequences of units by using the following distance metric:

$$D_2(i,j) = \sum_{k=0}^{L} D_1(i+k, j+k). \tag{11}$$

One can think of this as choosing a segment of the database unit curve, as traced out by the feature-space points $\{\mathbf{f}_j, \mathbf{f}_{j+1}, ..., \mathbf{f}_{j+L}\}$, that is closest to the warped target curve $\{\tilde{\mathbf{f}}_i, \tilde{\mathbf{f}}_{i+1}, ..., \tilde{\mathbf{f}}_{i+L}\}$). This tends to select more consecutive sequences of database units, and thus the resulting synthesized signal exhibits more of the original recording's temporal structure. In our examples, we use L values from 5 to 10 depending on material.

Avoiding large jumps: One issue that arises is that sometimes a database unit may be very far away in the D_1 metric, but close in the D_2 metric, resulting in undesirable "jump" artifacts which are difficult to blend. To avoid these cases, we limit our search to units u_j that are within a distance d_{max} of the warped target unit by using the modified distance function,

$$D_3(i,j) = \begin{cases} D_2(i,j) & \text{if } D_1(i,j) \le d_{max} \\ \infty & \text{otherwise} \end{cases} \tag{12}$$

In all our examples, $d_{max} = 2.0$, except for the corduroy model where $d_{max} = 3.0$. Using D_3 ensures that the unit u_j chosen by the distance $D_2(i,j)$ will not be too far in terms of $D_1(i,j)$, and it also serves as an optimization, since we do not have to compute $D_2(i,j)$ for every database unit; a similar approach is taken by Pullen and Bregler [2002]. On the off chance that no database units are within range, we fall back to the simple D_1 distance.

Avoiding repetition: Rapid repetition of selected units should be discouraged, since even repeating a unit once can produce a noticeable buzzing artifact. We explicitly avoid this by giving every unit a "cool-off period" L_c: if a unit u_j is selected, it is not allowed to be selected for the next L_c target units. In all our examples, $L_c = 20$.

Algorithm 1: Match database units $j = J(i)$ to target units i

```
begin
    // Compute distance table T
    for all pairs (i, j) do
        T(i, j) ← D3(i, j)
    // Fix up target units far from DB
    Far ← {i | T(i, j) = ∞ ∀j}
    for all pairs (i, j), i ∈ Far do
        T(i, j) = D1(i, j)
    // Compute matching J
    for each target unit i do
        J(i) = arg min_j T(i, j)
        // Enforce cool-off
        for k ← 1 to L_c do
            T(i + k, J(i)) = ∞
    return J
```

Concatenating units: Simply concatenating selected source units can produce many inter-unit discontinuities. Large C^0 discontinuities in an audio signal are heard as undesirable "crackles." To ensure that each unit starts and ends at a zero crossing, we can extend each unit boundary to an adjacent zero-crossing, and blend the extra samples using an overlap-add to avoid making the signal longer than the original target signal. Zero-crossings can be sparse for signals with strong low-frequency content, so we apply a 5th-order Butterworth high-pass filter with a 100 Hz cutoff frequency. This filter does not noticeably affect the synthesis quality, and it makes for a simple solution to large C^0 discontinuities.

5 Results

We now describe our results, but please watch and listen to the accompanying video with stereo headphones to hear the sonified animations. Figure 12 shows rendered stills from all our example animations. Parameters related to specific animations and materials are given in Table 1, and timings in Table 2. Parameters and timings specific to CSS warps and unit selection are given in Table 3.

Implementation Details: The commercially available cloth simulator *SyFlex* was used along with *Maya 2009* for all our examples. They were simulated and rendered at 60 FPS, and the simulation times ranged from half an hour up to 4 hours for our longest and most challenging examples. Simulations were done with two Intel Xeon X5570 processors (2.9 GHz, 8 cores total) using 16 threads. Specific timings for other parts of the pipeline are given in Table 2, but the cloth simulation was typically the dominant job. Motion-capture data for character animations are from the CMU Motion Capture Database. Geometry for the windbreaker jacket and pants are from *Poser 6*.

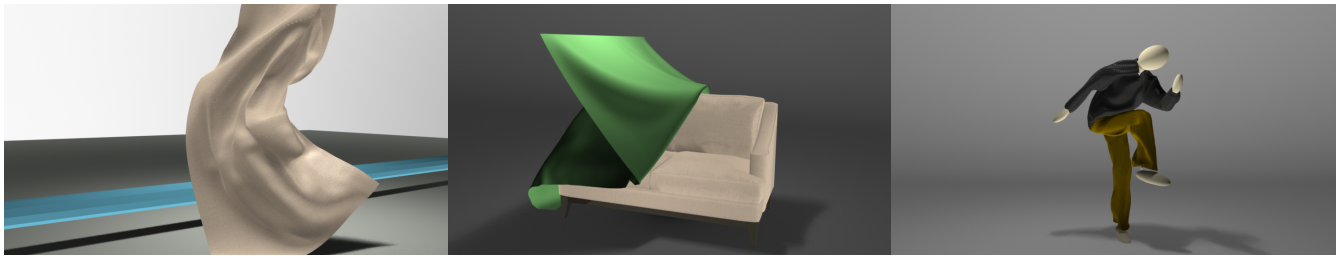
We record and synthesize all audio signals at 96 kHz. For all our examples except the binaural one, we essentially treat each stereo channel separately all the way through the pipeline. The results are different due to the use of a head-related transfer function, and they are combined in stereo in the video. The warps for the cotton and polyester sheets were trained using manual correspondences that covered both channels, as they were sufficiently different, but it sufficed to train only on the left channel for the character examples as the calibration animation is similar for both channels.

Lastly, our simulated examples often had contact speeds much higher than what we could measure in the friction sound experiment. In order to fully and evenly demonstrate the full range of friction sounds, we scaled down the speeds to keep them in range. For the character examples, due to the difficulty of simulating the stiff and light windbreaker material, we further applied a non-linear function to the sliding speeds in order to subdue lower–speed contacts that were causing excessive noise: $\bar{s}_v(t) = (s_v(t)/4)^{1.5}$. For speeds that were below our measured models, we quadratically scale down the lowest-speed spectrum $\mathcal{A}_0$ towards zero: $\mathcal{A}(f,s) = (s/s_0)^2 \mathcal{A}_0(f)$ for $s < s_0$.

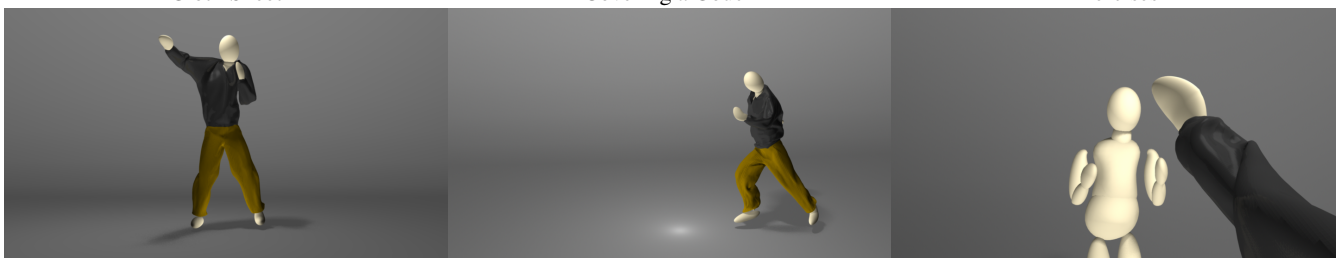
EXAMPLE (Cotton Sheet): This calibration animation exercises a cotton sheet by dragging it on the floor, picking it up, dropping it, and dragging it over a cylinder. Manual correspondences were specified at various points of crumpling and sliding, and particular care was taken to make sure the sliding motions produced no crumpling. The floor and cylinder were treated as being covered in cotton as well, as we do not handle friction sounds between multiple material types.

EXAMPLE (Cotton Couch): A sheet is draped over a cotton-lined couch, peeled off, and then dragged over again rapidly. It uses the warping from the Cotton Sheet example, so no manual intervention was required for CSS. Notice the characteristic crumpling at the end when the sheet quickly slips off the couch.

EXAMPLE (Polyester Sheet & Couch): We used the same animations as the previous two examples, except we synthesize them as polyester. The resulting sound is noticeably different, with higher pitched narrow-band content, as is characteristic of light polyester. When the sheet quickly slips off the couch at the end of the couch



Cloth Sheet | Covering a Couch | Exercises



Boxing | Jogging | First-Person Boxing

Figure 12: Result Animations: *Please view the accompanying video with stereo headphones to hear the sonified animations.*

example, the sliding noise quickly shifts up in pitch, giving a distinct "zip" effect.

EXAMPLE (Exercising): A motion-captured character goes through a variety of exercises and stretches while wearing a nylon windbreaker and corduroy pants. This serves as the calibration animation for these materials, so we manually tuned a warping and reused it for the next two examples (Jogging and Boxing). Some of the motion was edited in order to cover enough of the feature-space, such as the second half of the in-place jogging being sped up, so some parts may look artificial. It should be noted that we do not use the crumpling histogram match from this for the next two examples, as the crumpling energy distributions were too different.

EXAMPLE (Jogging): A character jogs back and forth a few times to demonstrate spatialization effects (HRTF [Brown and Duda 1998] and distance falloff). CSS automatically selects quieter source units for quieter target units, so the spatialization is preserved. Some jog cycles do not produce the expected "zip" of the corduroy pants due to the fact that the legs are often separated in the motion capture data.

EXAMPLE (Boxing): A character "shadow boxes," throwing a variety of punches. This demonstrates the strong synchronization between the synthesized sound and the animation. The corduroy pants are relatively quiet due to the boxer's wide stance.

EXAMPLE (First-person Boxing): The same boxing animation as in the previous example, but we position the camera and the listening position on the head of the character. In order to capture the sound of actually wearing a windbreaker—which is very different from hearing a windbreaker from a distance—we recorded a different database using the binaural microphones. Notice subtle spatialization effects in the final CSS result. The corduroy pants are not included in this sound.

6 Conclusion

We have presented a data-driven method for automatic concatenative synthesis of sounds for 3D cloth animations. We focus on two specific sound-producing phenomena, friction and crumpling, and we have demonstrated that these are sufficient for a variety of animated cloth scenarios. Our data-driven method is computationally efficient and requires a reasonable amount of recorded data and human intervention. This makes it practical for pre-rendered movies, and it has potential to be fast enough for interactive virtual environments in the future. Although it is difficult to make precise, objective measurements of accuracy, our results do compare favorably to actual recordings.

Table 1: Example Parameters: *The parameters used for the windbreaker in "boxing" were also used for "first-person boxing."*

Cloth Type	Animation	Collision radius r	s_{thresh}	E_{min}
Cotton	Sheet	0.05	0.02	4878
Cotton	Couch	0.2	0.02	4878
Polyester	Sheet	0.05	0.02	528
Polyester	Couch	0.2	0.02	528
Windbreaker	Exercises	0.05	0.0	98902
Windbreaker	Boxing	0.05	1.0	172875
Windbreaker	Jogging	0.05	0.0	906402
Corduroy	Exercises	0.05	2.0	-
Corduroy	Boxing	0.05	1.0	-
Corduroy	Jogging	0.05	2.0	-

Limitations & Future Work: Our focus on friction and crumpling sounds is motivated by the observation that these two sound sources dominate acoustic emissions arising from character clothing motions, but for high speed cloth animations, such as a flag waving in the wind or the whipping of bed sheet, other emissions due to increased tension and impacts become important. One potential approach is to do low-resolution simulations of such motions to get these tension-based sounds, where the cloth momentarily acts like a vibrating membrane, and add this to the target signal.

Our current pipeline does not allow for interactions between multiple types of cloth. Our features are not adequate for distinguishing between, for example, cotton contacting cotton versus cotton contacting polyester. More sophisticated features may be able to distinguish the two, allowing the target signal to contain a mix of various cloth types.

The interface for manual feature correspondences could be improved to be more intuitive. Currently, depending on the variety of sounds involved in a given example, finding a good set of manual correspondences can take many iterations of trial and error, some-

Table 2: Example Timings: *All timings were done on an 8 core 2.93GHz Intel Xeon processor.*

Cloth Type	Animation	Friction Motion Analysis (s)	Friction Synthesis (s)	Crumpling Motion Analysis (s)	Crumpling Synthesis (s)	Target Unit Feature Comp. (s)	Unit Selection (s)
Cotton	Sheet	37	135	12	21	73	25
Cotton	Couch	104	310	16	13	132	48
Polyester	Sheet	37	295	12	66	74	33
Polyester	Couch	104	455	16	16	132	62
Windbreaker	Exercises	315	94	62	35	152	41
Windbreaker	Boxing	766	255	120	101	472	197
Windbreaker	Jogging	373	86	60	52	199	55
Corduroy	Exercises	55	90	-	-	45	5
Corduroy	Boxing	124	75	-	-	79	24
Corduroy	Jogging	58	80	-	-	37	6
Windbreaker	FP Boxing	766	274	120	94	446	65

Table 3: CSS Model Parameters and Timings: *Each calibrated CSS model consists of a TPS warp fit to manual correspondences along with unit selection parameters. The animation listed is the calibration animation, and the same parameters were used for other animations that reuse the CSS model. The number of correspondence units is effectively the number of TPS nodes used for warping, and time to perform TPS fitting was negligible. All manual correspondences were clips of length 0.1 or 0.2 seconds long. Gaussian filter widths are given in number of sound units, and the standard deviation was a quarter of the width.*

Cloth Type	Animation	# Manual corr.	# Corr. units	TPS λ	L	Gaussian widths	DB Length (m:ss)	DB Feature Comp. (s)
Cotton	Sheet	14	419	1.0e2	5	20, 50, 10	3:45	1317
Polyester	Sheet	9	280	1.0e5	10	10, 50, 20	4:05	1392
Windbreaker	Exercises	5	187	1.0e5	5	20, 30, 10	3:13	1101
Corduroy	Exercises	3	15	0.0	10	20, 30, 10	2:41	235
Windbreaker	FP Boxing	8	208	1.0e5	5	20, 30, 10	1:32	1051

times up to 5. We conservatively estimate that each warping function in our examples took about 1-3 hours to tune, including manual correspondences and synthesis of the calibration animation to assess the warping with various TPS λ values. Specifying the correspondences only takes about 5 to 15 minutes per iteration (please see the supplemental video for a typical session), and the rest of the time is for the CSS pipeline. The time to specify correspondences and total number of iterations may vary depending on the experience level of the user. Future work will focus on providing the user with rapid and intuitive feedback as to which correspondences are causing undesirable synthesis results, and the system should also suggest correspondences based on analysis of the feature distributions.

Other potential areas for future research include acoustic transfer, such as occlusion of sound emissions by large bodies. It is unclear how to approach the problem of acoustic transfer around highly deformable cloth, and it is also unclear how much it matters. Also, we fully simulate clothing in our character examples, but it may be desirable to produce sound for skinned cloth models that lack proper sliding and crumpling events. Lastly, we use a local greedy unit selection algorithm for its potential to be used in online applications, but it is likely that global optimization techniques would provide higher quality results for prerendered movies.

Acknowledgments

We would like to thank anonymous reviewers for helpful feedback, Sean Chen for the measurement device circuitry, the Clark Hall Machine Shop, Carol Krumhansl for the use of her sound isolation room, Taylan Cihan for early recording assistance, Tianyu Wang for signal processing discussions, and Noah Snavely for thoughts on shape matching. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. DLJ acknowledges early discussions with Dinesh Pai and Chris Twigg on cloth sounds. We acknowledge funding and support from the National Science Foundation (CAREER-0430528, HCC-0905506, IIS-0905506), fellowships from the Alfred P. Sloan Foundation and the John Simon Guggenheim Memorial Foundation, and donations from Pixar and Autodesk. This research was conducted in conjunction with the Intel Science and Technology Center–Visual Computing. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or others.

References

BARAFF, D., AND WITKIN, A. P. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, 43–54.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Trans. on 24*, 4 (April), 509 –522.

BILBAO, S. 2009. *Numerical Sound Synthesis*. Wiley Online Library.

BRIDSON, R., FEDKIW, R. P., AND ANDERSON, J. 2002. Robust treatment of collisions, contact, and friction for cloth animation. *ACM Transactions on Graphics 21*, 3 (July), 594–603.

BROWN, C., AND DUDA, R. 1998. A structural model for binaural sound synthesis. *Speech and Audio Processing, IEEE Transactions on 6*, 5 (sep), 476 –488.

CARDLE, M., BROOKS, S., BAR-JOSEPH, Z., AND ROBINSON, P. 2003. Sound-by-numbers: motion-driven sound synthesis. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '03, 349–356.

CHADWICK, J. N., AND JAMES, D. L. 2011. Animating fire with sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011) 30*, 4 (Aug.).

CHADWICK, J. N., AN, S. S., AND JAMES, D. L. 2009. Harmonic shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics 28*, 5 (Dec.), 119:1–119:10.

CHO, G., CASALI, J., AND YI, E. 2001. Effect of fabric sound and touch on human subjective sensation. *Fibers and Polymers 2*, 196–202.

CHO, G., KIM, C., CHO, J., AND HA, J. 2005. Physiological signal analyses of frictional sound by structural parameters of warp knitted fabrics. *Fibers and Polymers 6*, 89–94.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. *ACM Transactions on Graphics 21*, 3 (July), 604–611.

COOK, P. 2002. *Real Sound Synthesis for Interactive Applications*. A.K. Peters.

COURSHESNES, M., VOLINO, P., AND MAGNENAT-THALMANN, N. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 137–144.

DUBNOV, S., BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2002. Synthesizing sound textures through wavelet tree learning. *Computer Graphics and Applications, IEEE 22*, 4, 38–48.

FONTANA, F., AND BRESIN, R. 2003. Physics-based sound synthesis and control: crushing, walking and running by crumpling sounds. In *Proc. Colloquium on Musical Informatics*, 109–114.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 229–238.

HOULE, P., AND SETHNA, J. 1996. Acoustic emission from crumpling paper. *Physical Review E 54*, 1, 278.

HUANG, G., METAXAS, D., AND GOVINDARAJ, M. 2003. Feel the "fabric": an audio-haptic interface. In *2003 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 52–61.

HUNT, A., AND BLACK, A. 1996. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 1, IEEE, 373–376.

JAMES, D. L., BARBIČ, J., AND PAI, D. K. 2006. Precomputed Acoustic Transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics 25*, 3 (July), 987–995.

KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM Transactions on Graphics 27*, 3 (Aug.), 65:1–65:9.

KARPLUS, K., AND STRONG, A. 1983. Digital Synthesis of Plucked-String and Drum Timbres. *Computer Music Journal 7*, 2, 43–55.

MARELLI, D., ARAMAKI, M., KRONLAND-MARTINET, R., AND VERRON, C. 2010. Time–frequency synthesis of noisy sounds with narrow spectral components. *Audio, Speech, and Language Processing, IEEE Transactions on 18*, 8, 1929–1940.

MATUSIK, W., ZWICKER, M., AND DURAND, F. 2005. Texture design using a simplicial complex of morphable textures. *ACM Transactions on Graphics 24*, 3 (Aug.), 787–794.

O'BRIEN, J. F., COOK, P. R., AND ESSL, G. 2001. Synthesizing sounds from physically based motion. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 529–536.

O'BRIEN, J. F., SHEN, C., AND GATCHALIAN, C. M. 2002. Synthesizing sounds from rigid-body simulations. In *ACM SIGGRAPH Symposium on Computer Animation*, 175–181.

PELTOLA, L., ERKUT, C., COOK, P. R., AND VALIMAKI, V. 2007. Synthesis of hand clapping sounds. *Audio, Speech, and Language Processing, IEEE Transactions on 15*, 3, 1021 –1029.

PICARD, C., TSINGOS, N., AND FAURE, F. 2009. Retargetting example sounds to interactive physics-driven animations. In *AES 35th International Conference on Audio for Games*.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics 21*, 3 (July), 501–508.

RABINER, L., AND JUANG, B. H. 1993. *Fundamentals of Speech Recognition*, united states ed. Prentice Hall, Apr.

RAGHUVANSHI, N., AND LIN, M. C. 2006. Interactive Sound Synthesis for Large Scale Environments. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM Press, New York, NY, USA, 101–108.

ROADS, C. 2004. *Microsound*. The MIT Press.

RODET, X., DEPALL, P., RODET, X., AND DEPALLE, P. 1992. Spectral envelopes and inverse FFT synthesis. In *Proceedings of the 93rd Audio Engineering Society Convention*.

SCHWARZ, D. 2004. *Data-Driven Concatenative Sound Synthesis*. PhD thesis, Ircam, Centre Pompidou, University of Paris 6–Pierre et Marie Curie.

SERRA, X., AND SMITH, JULIUS, I. 1990. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal 14*, 4, pp. 12–24.

STROBL, G., ECKEL, G., ROCCHESSO, D., AND LE GRAZIE, S. 2006. Sound texture modeling: A survey. In *Proceedings of the 2006 Sound and Music Computing (SMC) International Conference*, 61–5.

TAKALA, T., AND HAHN, J. 1992. Sound rendering. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, 211–220.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, 205–214.

VAN DEN DOEL, K., AND PAI, D. K. 1996. Synthesis of shape dependent sounds with physical modeling. In *Intl Conf. on Auditory Display*.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. FoleyAutomatic: Physically Based Sound Effects for Interactive Simulation and Animation. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 537–544.

WAHBA, G. 1990. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, Philadephia, PA.

ZHENG, C., AND JAMES, D. L. 2010. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics 29*, 4 (July), 69:1–69:13.