UC Berkeley
UC Berkeley Previously Published Works

Title
Near-exhaustive precomputation of secondary cloth effects

Permalink
https://escholarship.org/uc/item/4113j3t6

Journal
ACM Transactions on Graphics, 32(4)

ISSN
0730-0301

Authors
Kim, Doyub
Koh, Woojong
Narain, Rahul
et al.

Publication Date
2013-07-21

DOI
10.1145/2461912.2462020

Supplemental Material
https://escholarship.org/uc/item/4113j3t6#supplemental

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Near-exhaustive Precomputation of Secondary Cloth Effects

Doyub Kim[1] Woojong Koh[2] Rahul Narain[2] Kayvon Fatahalian[1] Adrien Treuille[1] James F. O'Brien[2]

[1]Carnegie Mellon University [2]University of California, Berkeley

Figure 1: *Our system animates this detailed cloth motion at over 70 FPS with a run-time memory footprint of only 66 MB. We achieve high quality and high performance by compressing over 33 GB of data generated during 4,554 CPU-hours of off-line simulation into a 99,352 frame secondary motion graph that tabulates the cloth dynamics.*

Abstract

The central argument against data-driven methods in computer graphics rests on the curse of dimensionality: it is intractable to precompute "everything" about a complex space. In this paper, we challenge that assumption by using several thousand CPU-hours to perform a massive exploration of the space of secondary clothing effects on a character animated through a large motion graph. Our system continually explores the phase space of cloth dynamics, incrementally constructing a *secondary cloth motion graph* that captures the dynamics of the system. We find that it is possible to sample the dynamical space to a low visual error tolerance and that secondary motion graphs containing tens of gigabytes of raw mesh data can be compressed down to only tens of megabytes. These results allow us to capture the effect of high-resolution, off-line cloth simulation for a rich space of character motion and deliver it efficiently as part of an interactive application.

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

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

Links: DL PDF VIDEO WEB

1 Introduction

Data-driven techniques have enabled real-time animation of stunningly complex phenomena that are either too expensive to simulate in real-time (such as the folds and wrinkles in high-resolution cloth [Kavan et al. 2011; Wang et al. 2010]) or for which we lack good models (such as human motion). However, the central argument against these precomputation-based approaches rests on the curse of dimensionality: it is impossible to capture "everything" because each additional simulation condition exponentially explodes the dynamical space. For this reason, virtually all previous work in data-driven animation has studied phenomena in a very controlled settings with few excitation modes, and under the strict assumption that the phenomenon will periodically return to a single, rest configuration [James and Pai 2002; James and Fatahalian 2003; de Aguiar et al. 2010; Guan et al. 2012].

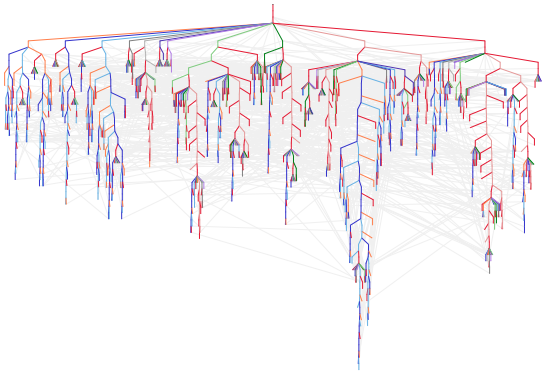


Figure 2: *Visualization of **Robe**'s 99,352-frame secondary motion graph. Cloth trajectories are colored by their corresponding character motion. Edges created as a result of dead-end merges (Section 3) are shown in light gray.*

In light of rapid growth in the availability of low-cost, massive-scale computing capability we believe that it is time to revisit this assumption. While it might not be tractable to capture *everything* about a complex dynamical system, it may be possible to capture *almost everything important*. This mirrors a growing trend in computer science where researchers studying theoretically infinite spaces like machine translation have captured "almost everything" about translation, for example, simply by using a sufficiently large data corpus [Halevy et al. 2009].

In this paper, we focus on the precomputation of secondary clothing effects for a character animated through a finite motion graph. We introduce the notion of a *secondary motion graph*: for each state in the primary motion graph (in this case, the character's pose) there may be many corresponding states in the secondary motion graph (in this case, configurations of clothing on the body). Because cloth is a dynamical system where the state of the cloth depends on previous cloth states, not just the body's pose, the secondary motion graph can be vastly more complex than the primary motion graph.

We report our findings from performing a massive exploration of the secondary motion graph space. In contrast to previous work on precomputation, we simulate significantly more data to build a large-scale portrait of the phase space of the dynamics. Our primary

From the conference proceedings of ACM SIGGRAPH 2013.
Appearing in ACM Transaction on Graphics Vol. 32, No. 4.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ACM SIGGRAPH, 2013, Anaheim, CA
© Copyright ACM 2013

example represents over 55 minutes of simulated cloth animation generated over 4,500 CPU-hours of computation. The scale of our exploration allows us to observe *bifurcations* in the dynamics where persistent changes in the cloth configuration create new subgraphs that remain largely disjoint from previously computed states. We show that even for large motion graphs, it is possible to explore the secondary dynamical space to low visual error tolerance — essentially, to precompute almost everything important — and that the resulting data can be compressed to a reasonably small memory footprint. On a practical level, these results allow us to capture the effect of high-resolution, off-line cloth simulation for a rich space of character motions and deliver it in real time on modest hardware as part of an interactive application.

In contrast to many data-driven techniques, our model provides guaranteed bounds on the error of the approximation for the precomputed model. Error due to insufficient sampling only manifests at a sparse set of frames in the database. As a result it is possible to verify the error entirely and even manually inspect these error frames. In addition, our system is designed to support "continual exploration" of the dynamical space wherein an ever-enlarging portrait of the phase space is constructed. In contrast, previous precomputation approaches typically require careful set up of the exact scenarios to be precomputed.

2 Related Work

The use of computer simulation to generate cloth motion for animation has been an active research topic for over a quarter of a century [Terzopoulos et al. 1987]. During that time key problems related to modeling the dynamics of cloth and for handling collisions have been addressed [Baraff and Witkin 1998; Bridson et al. 2002; Choi and Ko 2002; Bridson et al. 2003]. Summaries covering much of development of cloth simulation methods can be found in the survey articles of Nealen et al. [2006] and Thomaszewski et al. [2007]. Many current methods for simulating cloth can produce highly realistic results that can be difficult to distinguish from reality [Müller and Chentanez 2010; Narain et al. 2012]. The reality of these methods can be enhanced by using measured material properties [Wang et al. 2010; Miguel et al. 2012], or even simulating physics at the level of individual yarns [Kaldor et al. 2010].

Unfortunately, the computational cost of realistic high-resolution cloth simulation currently precludes its use in interactive applications. Graphics researchers have addressed this issue using a variety of methods, many of which also make use of precomputed data.

One common approach is to run low resolution simulation and then add extra wrinkle detail based on previously computed high-resolution simulations [Wang et al. 2010; Kavan et al. 2011; Feng et al. 2010], recordings of real cloth [Popa et al. 2009; Hilsmann and Eisert 2012], or a simplified physical model [Müller and Chentanez 2010; Rohmer et al. 2010]. While these approaches dramatically increase cloth realism with minimal computational overhead, the resulting motion can still reflect the low-resolution of the underlying simulation mesh.

Researchers have also looked into model reduction methods to fully capture complex systems such as fluids [Treuille et al. 2006; Barbič and Popović 2008; Wicke et al. 2009] and deformable objects [James and Pai 2002; Barbič and James 2005; An et al. 2008] in a reduced order model. This approach can lead to significant speedups, but has difficulty capturing discontinuities and places other restrictions on the underlying dynamics. Furthermore, such data-driven approaches often require that the simulated motion hews closely to the precomputed data with unpredictable error if the system strays too far from the captured model. This paper explores a different approach, using a much simpler graph-based model to tabulate the dynamics while emphasizing massive precomputation to capture a large portion of the cloth's phase space.

Rather than reduce the dynamics of clothing on a human character, the stable spaces technique of de Aguiar et al. [2010] eschews run-time simulation entirely and instead learns a quasi-linear model for the dynamics from black-box simulation data. The learned model approximates cloth motion based on body pose (echoing data-driven skinning approaches) and the recent history of the cloth.

In contrast to de Aguiar's work, James and Fatahalian [2003] explicitly tabulate the (arbitrarily non-linear) dynamics of a deformable system. Real-time cloth animation is achieved by navigating a database of cloth trajectories residing in the precomputed subspace. Our work is similar in spirit to theirs but differs in two key ways. First, rather than drive the system using a small palette of simple impulses, we represent a much richer space of external cloth forces using a character motion graph. Second, the sheer scale of our state-space sampling process allows us to tabulate much more complex cloth behaviors, including spaces that exhibit bifurcations and cloth that does not return to a single rest state. Our work addresses the subspace sampling and data-compression challenges of distilling large amounts of precomputed data into a representation that delivers high-quality cloth animation in an interactive system.

A key benefit of our approach is that it naturally models bifurcations in the phase space, a key challenge for precomputation methods. Twigg and James have exploited dynamical bifurcations (arising from collisions) for control in graphics. One of their methods explores contact ambiguity to create animations which converge to a desired final state [Twigg and James 2008]. Another visualizes the bifurcating dynamics of colliding objects as an interaction paradigm for control [Twigg and James 2007]. Our work similarly studies large-scale phase spaces including bifurcations, but our goal is different: rather than seeking control, we view bifurcations as a modeling challenge for data-driven interactive simulation

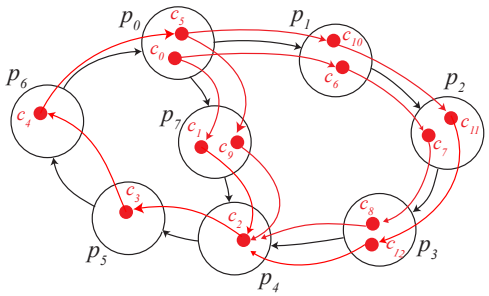
3 The Secondary Motion Graph

We pursue an entirely data-driven system for synthesizing real-time, high-resolution motion of clothing on a human character. Our approach relies on an extensive sampling of simulated cloth motion that results from animating a character through a large motion graph. As new cloth motions are generated they are analyzed and compressed into a compact representation enabling low-cost run-time use. We now describe our cloth motion representation and the techniques employed to sample, compress, and interactively play back precomputed cloth motion.

3.1 Secondary Graph Definition

We represent character movement and its associated secondary cloth dynamics using two motion graphs [Kovar et al. 2002; Arikan and Forsyth 2002]. The first graph, which we call the *primary graph*, is a standard motion graph describing movement of the character. Our graph formulation represents individual character poses (rather than motion clips) as graph nodes p_i. Thus, a graph edge from p_i to p_j indicates p_j can directly follow p_i in animation. A tiny example of a primary motion graph with seven states is illustrated in black in Figure 3.

We augment the primary motion graph with a *secondary cloth motion graph* that tabulates the enormous (potentially infinite) space of cloth dynamics. Each node in the secondary graph (13 red states in Figure 3) represents a cloth pose c_i and it is associated with exactly one primary graph node $P(c_i)$. That is, the character is in state $P(c_i)$ when the cloth assumes pose c_i. Extensive sampling of



Cloth state	Observed Character Pose Transition	New Cloth State $C(c_i, p_j \rightarrow p_k)$
c_0	$p_0 \rightarrow p_7$	c_1
c_0	$p_0 \rightarrow p_1$	c_6
c_5	$p_0 \rightarrow p_7$	c_9
c_5	$p_0 \rightarrow p_1$	c_{10}

Figure 3: *Top: As clothing can assume many positions for each character pose, many states in the secondary cloth motion graph (red) correspond to each primary graph (black) state. Bottom: Traversal of the primary graph (character animation) triggers corresponding traversal of the secondary graph (cloth animation). Motion of the cloth in response to character motion is defined entirely by the transition table $C(c_i, p_j \rightarrow p_k)$.*

the cloth phase space will result in a secondary motion graph that is significantly larger than the primary graph. Thus, as shown in Figure 3, many secondary graph states may be associated with each node in the primary graph.

During playback, traversal of the primary graph is governed by the application's character controller as is common in interactive applications today. To produce cloth motion, each primary graph transition from p_j to p_k also triggers a corresponding secondary graph transition from current cloth pose c_i to $C(c_i, p_j \rightarrow p_k)$. A table enumerating secondary graph transitions for cloth states c_0 and c_5 is provided in the figure.

The secondary graph and its associated transition table $C(c_i, p_j \rightarrow p_k)$ describe the cloth's motion in response to external forces applied by the character. Together, their function is analogous to the database of impulse response functions described by James and Fatahalian [2003]. By representing external forces as transitions in the primary motion graph we are able to encode a significantly richer space of forces than James and Fatahalian's impulse palette.

3.2 Graph Exploration

Due to its dynamic motion, clothing may assume an arbitrarily large number of configurations as the character continuously animates through the primary graph. Therefore, it is likely intractable to tabulate the secondary motion graph in its entirety. Instead, we incrementally build the secondary motion graph by repeatedly traversing segments of the primary graph and computing the corresponding cloth motion using a black-box off-line cloth simulator [Narain et al. 2012]. Our system is specifically designed to be able to "sample forever," exploring for undiscovered motions, while continuously maintaining a valid secondary motion graph for immediate interactive use.

Graph Initialization. We initialize the secondary graph by exploring each primary graph edge once via breadth-first traversal starting from an arbitrarily chosen initial character pose. The clothing is placed on this initial pose and traversing the graph requires forward simulation of the clothing over the traversed character poses.

RemoveDeadend(c_{end}):

($c_{similar}$, *err*) = FindMostSimilar(c_{end})

// merge c_{end} into $c_{similar}$
foreach edge e incoming to c_{end} **do**:
    modify e to point to $c_{similar}$
remove c_{end} from graph

workQueue.AddJob(*err*, (c_{end}, $c_{similar}$))

ExpandGraph():

do forever:
    (c_{end}, $c_{similar}$) = *workQueue*.LargestErrorJob()

    // revert the merge of c_{end} into $c_{similar}$
    re-insert c_{end} into graph
    foreach edge e incoming to $c_{similar}$ **do**:
        if e resulted from prior merge with c_{end}:
            modify e to point to c_{end}

    newSubtree = SimulateCloth(c_{end})
    foreach new dead-end c_{new} in *newSubtree* **do**:
        RemoveDeadend(c_{new})

Figure 4-left illustrates exploration beginning at node p_0 with initial cloth state c_0. The result is a tree of cloth trajectories (shown in red) produced from simulation. (The tree is re-drawn for clarity in the bottom-left of the figure.) Each path from the tree's root c_0 to a leaf c_i represents a simulated cloth motion sequence corresponding character animation from initial pose $P(c_0) = p_0$ to $P(c_i)$.

At this point, the leaves of the tree constitute dead-ends in the secondary motion graph. We remove each dead-end state by merging it with the most similar interior cloth state that shares *the same primary graph state* (see function RemoveDeadend()). Notice that in the center panel of Figure 4, dead-end nodes c_5 and c_9 have been merged into c_0 and c_2 respectively. (Removed nodes c_5 and c_9 are shown in gray in the secondary graph.) Edges that previously transitioned to dead-end nodes now transition into the target of the merges (shown in blue). Once all dead-ends have been removed, graph initialization is complete. The secondary graph now provides a cloth animation result for any sequence of character motion and thus represents a coarse approximation to the full space of cloth dynamics. While red edges in the figure correspond to actual simulated trajectories, blue edges constitute transitions that *approximate* the real cloth dynamics. When the approximation is poor (that is, when no interior node in the graph is similar to the dead-end state), further sampling of cloth phase space is needed to reduce this error and avoid low-quality motion during playback.

Endless Graph Expansion. After each dead-end merge operation, we store both the (pre-merge) dead-end cloth graph state and its associated (post-merge) edge approximation error in a priority work queue for further sampling. To reduce cloth motion error, we extensively refine the secondary-motion graph using the process given by function ExpandGraph(). In each expansion step we extract the largest error dead-end c_{max} from the priority queue and reinsert c_{max} into the graph by reverting the prior graph merge operation that eliminated it. We then simulate further cloth motion beginning from c_{max} (and character pose $P(c_{max})$). Further simulation samples a previously unexplored region of the cloth phase space and produces additional, new dead-ends in the secondary graph. These new dead-ends are eliminated as described before via additional merges.

Figure 4-right shows the results of performing one step of graph exploration beginning with enqueued dead-end state c_5. The addi-

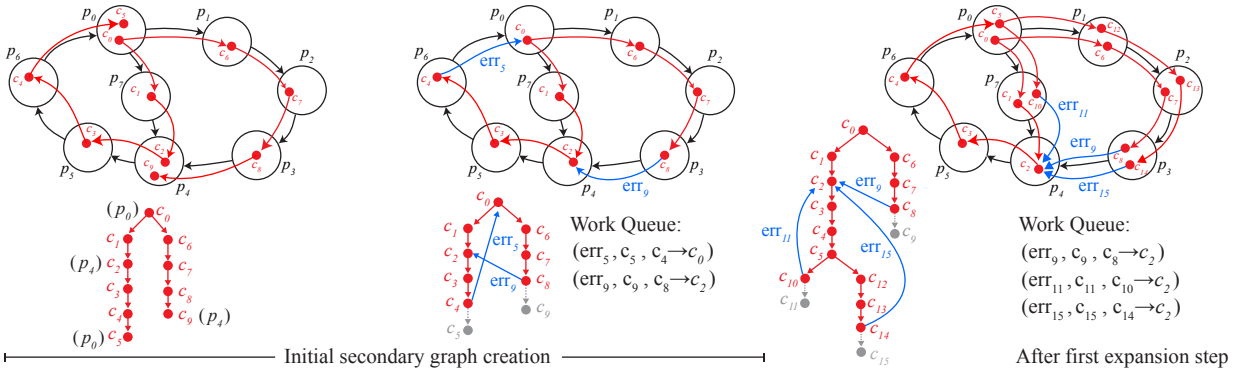


Figure 4: *Left: Secondary graph initialization begins by sampling cloth motion once for each transition in the character motion graph. (Secondary cloth states are shown in red.) Center: Dead-end states (c_5 and c_9) are removed by merging them with similar interior graph nodes. The resulting edges (blue) correspond to motions that only approximate actual cloth dynamics. Right: Error in the secondary graph is reduced by continually replacing the highest-error edges with the results of additional simulation.*

tional simulation results in two new dead-ends, both of which are merged into c_2. The process of secondary graph refinement continues repeatedly, eliminating the largest physics errors encountered as the cloth phase space is explored while always maintaining a dead-end-free secondary motion graph. Although the simulations carried out in each new expansion step depend on prior results, expansion of the graph from each dead-end in the work queue is independent, making graph expansion embarrassingly parallel.

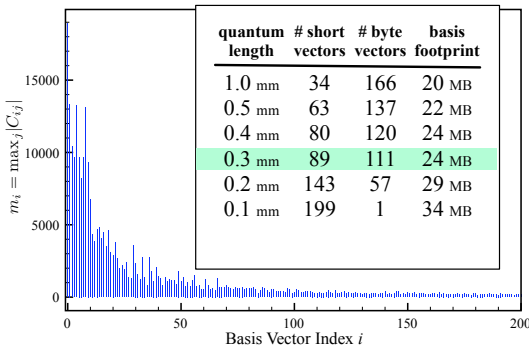
3.3 Refinements

Garment-customized error metric. Certain cloth transition errors are significantly more objectionable than others. To provide the system knowledge of the worst errors, we can customize the cloth-pose similarity metric for a particular garment or domain of motion. For example, in the case of the robe garment (discussed in Section 5) it is particularly objectionable for the cloth graph to contain back-links that take the hood from a position down on the character's shoulders to up on its head. We find that simple error metrics (e.g., L_1 or L_∞ error for vertex positions and velocities) do not identify such transitions as egregious, so we adopt a metric that classifies the hood's position for a cloth pose as up or down and assigns high error to merges that create implausible down-to-up transitions. As a result, these transitions rise to the head of the queue and the system quickly eliminates the error by exploring dynamics beginning from these high-error dead-ends.

Graph Blending. Motion error in the secondary graph occurs precisely at edges that result from dead-end merges (blue edges in Figure 4). To reduce visual artifacts caused by these discontinuities we diffuse their error over a sequence of frames (our implementation uses ten) by linearly blending trajectories just before and after the merged state.

4 Compression

After any step of secondary graph expansion we can compress and package the data for interactive visualization or integration into a game engine. Compression is crucial for interactive playback as our raw secondary-graph data consists of tens of thousands of frames of cloth meshes requiring hundreds of gigabytes of storage. However, as others have previously observed [Wang et al. 2010], clothing tends to form nearly repetitive patterns and there is substantial redundancy in these data. We have found that the raw secondary-graph data compresses remarkably well, and that compressed results have a small footprint and can be decompressed efficiently in real time. This compression is critical as modern games typically have tight memory budgets, particularly for secondary effects.



quantum length	# short vectors	# byte vectors	basis footprint
1.0 mm	34	166	20 MB
0.5 mm	63	137	22 MB
0.4 mm	80	120	24 MB
0.3 mm	89	111	24 MB
0.2 mm	143	57	29 MB
0.1 mm	199	1	34 MB

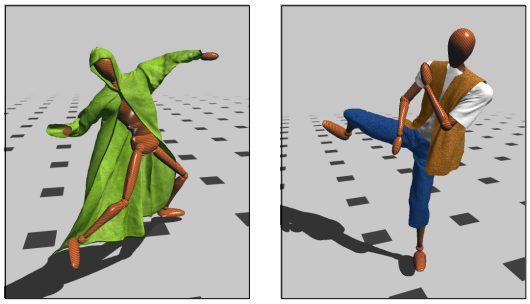
Figure 5: *Graph: maximum contribution to cloth mesh displacement by each basis vector (**Robe** secondary graph). Many vectors provide only a small contribution and thus are candidates for further compression. Inset: Quantization of basis vector values allows for compact representation. Quantization to three-hundred-micron tolerance yields results that are visually indistinguishable from the original data and permit basis representation using 16-bit and 8-bit values.*

We first reduce the dimensionality of the data set by performing an out-of-core SVD as described by James and Fatahalian [2003]. This step factors the cloth data matrix $D \in \mathbf{R}^{3v \times n}$ (containing n frames of v vertex positions) into a b-dimensional cloth basis $B \in \mathbf{R}^{3v \times b}$ and a matrix of trajectory coefficients $C \in \mathbf{R}^{b \times n}$ so that $D \approx BC$. We experimentally determined that with 200 bases ($b = 200$) the reconstructed cloth mesh is visually very similar to the original model. However in places where the cloth is in contact with the underlying body model, even small compression errors can cause a reversal in depth ordering causing clear rendering artifacts. We eliminate these artifacts during rendering using the depth offset technique of [de Aguiar et al. 2010] and by very slightly shrinking the body model.

We further observe that although fewer basis vectors are not adequate for maintaining good visual fidelity, individual basis vectors vary greatly in their geometric impact. We denote $m_i = \max_j |C_{ij}|$ as the maximum absolute contribution of each basis and find that m_i can vary by orders of magnitude across the basis (Figure 5). We achieve further compression by representing low impact basis vectors with fewer bits.

We re-scale the cloth basis by $\{m_i\}$ to produce $B' = BM$ and $C' = M^{-1}C$, where $M = \mathrm{diag}(m_1, \ldots, m_b)$. The re-scaled basis B' can be interpreted in length units of maximum vertex displace-

	Robe	Casual
Cloth vertex count	29,654	32,984
Average sim. time / frame (s)	165	243
Total simulated frames	99,352	27,545
Total CPU-hours	4,554	1,859
Uncompressed DB size (MB)	33,614	10,397
Compressed basis size (MB)	24	22
Compressed trajectories size (MB)	42	12
Total runtime footprint (MB)	66	34



Robe Casual

Table 1: *Secondary motion graph exploration and compression statistics for our two example garments.*

ment, and is now amenable to quantization. We found 300-micron quanta represent the coarsest possible quantization before visual artifacts become evident. At this tolerance, basis vectors are easily representable using 16-bit shorts and 8-bit bytes. In the case of our **Casual** demo scene (Section 5), 82% of basis vectors are representable using 8-bit values. The resulting compressed basis requires only 22 MB of storage. Similar quantization to 16-bit values is also performed on the trajectory coefficients. This simple compression scheme using two bit widths achieves high compression while enabling efficient decoding of trajectory data in matrix C' at run time.

5 Results

To evaluate our method, we conducted a massive exploration of cloth motion by animating a character through a motion graph constructed from the HDM05 motion capture data set [Müller et al. 2007]. Our primary motion graph contains 12 unique motion clips (totaling 3,115 frames). We selected clips featuring a variety of vigorous motions, including running, throwing, kicking, hopping, and performing cartwheels, to trigger highly dynamic cloth motion and to direct garments into a wide range of configurations. From this primary graph, we generate secondary motion graphs for two cloth setups: **Robe**, a one-piece, hooded robe, and **Casual**, which features three layered garments (including baggy pants and a loose-fitting vest). We used ARCSim[1], a high-quality, off-line cloth simulator, to compute cloth motion [Narain et al. 2012; Wang et al. 2011]. To accommodate our compression scheme we disabled adaptive remeshing and used a fixed mesh topology. On average, off-line frame simulation times for our two demos are 165 seconds and 243 seconds respectively. Although both demos feature garments of similar resolution, complex collision handling between the multiple cloth layers in **Casual** results in longer execution time.

[1] http://graphics.berkeley.edu/resources/ARCSim

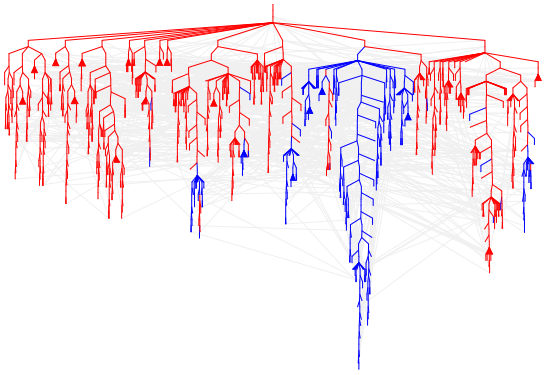


Figure 6: *Visualization of hood state in the **Robe** secondary graph. Red indicates the hood is up, blue incades the hood has fallen. Simulation begins from cloth pose with the hood up. Graph expansion uncovers motion sequences that cause the hood to fall, then it simultaneously explores these disjoint regions of the cloth phase space. Our method automatically explores both branches of this dynamical bifurcation in parallel.*

5.1 Large-scale Exploration

Using more than 6,400 CPU-hours of computation (on a cluster of machines provided by Intel) we generated over 99,000 frames of cloth animation (about 55 minutes of simulated motion, visualized directly in Figures 2 and 6) for **Robe** and 27,000 frames of animation (15 minutes) for **Casual**. Both secondary graphs are approximately an order-of-magnitude larger than the character motion graph. For brevity we focus on our findings from the **Robe** exploration in this section and refer the reader to the accompanying video to view results from **Casual**.

Simply put, we find that our massive computation *has* adequately sampled the phase space of cloth dynamics to good visual tolerance. We are able to generate pleasing cloth animations for arbitrary traversals of the primary graph. As evident in the accompanying video, paths through the secondary graph produce highly detailed, smooth cloth motion even after vigorous character actions, such as a cartwheel (Figure 1), that leave the cloth in a dramatically different pose than its starting state.

Although our final secondary graphs do represent high-quality motion, this was not true after early stages of exploration. Immediately following graph initialization, the secondary graph contains only one cloth pose for each primary character pose and thus resembles a kinematic approximation to cloth motion. As expected, this approximation was severely inadequate to produce realistic motion for our garments, confirming that the secondary cloth graph must be significantly more complex than the primary graph. In these early phases, large errors were clearly visible as the cloth transitioned from complex configurations back to configurations near its initial starting state.

Further exploration incrementally eliminates egregious motion errors and, more importantly, uncovers interesting structure in the secondary motion space, such as bifurcations. While our initial **Robe** configuration features the hood covering the character's head, we find that a few sequences of motion (e.g., walk → jumping-jack → walk, or walk → jog left → walk backward → jog right) cause the hood to fall. Figure 6 illustrates the complete **Robe** secondary graph with states colored according to the hood's state. (Red indicates the hood remains on the character's head, while blue indicates the hood has fallen off.) Our exploration finds five unique character motion sequences that cause the hood to fall. It then explores these distinct regions of the phase space in parallel. When the bifurcation is first encountered, there are very few secondary graph states with

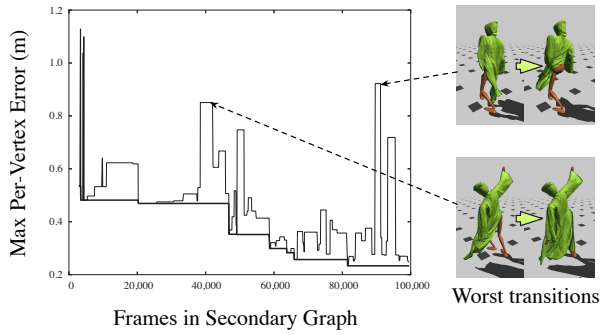


Figure 7: *The thinner line represents the maximum back-link error over time. The thicker line represents the lowest maximum error we can get from the database at certain time by excluding the additional data until it reduces the maximum error in the graph. At coarse scale, motion error steadily declines as the **Robe** state space is explored in detail. Novel behaviors in the space temporally increase error until they are sufficiently sampled.*

the hood down. As a result, dead-end merges often create transitions that take the hood from the down to the up state (there are no similar hood-down poses to merge with). However, our error metric (which classifies hood state) tags these implausible transitions with high error, so the system immediately dedicates compute resources to exploring these parts of the graph. Once the secondary graph has reached 99,000 frames in size, is free of merged transitions from hood down to hood up states. The graph is roughly partitioned into two subgraphs based on the hood state. The edges linking those subgraphs all correspond to correct simulated motion where the hood naturally falls down or falls over the character's head as he does a flip.

Figure 7 plots the evolution of L_∞ error (from a vector containing both vertex positions and time-scaled velocities) of the worst transition in the secondary graph. At coarse scale, error in the graph decreases as we explore the phase space in detail. However, there is significant fine-scale variation in error, which we attribute to two reasons. First, a simple distance metric can be a poor predictor of future error in the phase space. (Improving error metrics to consider potential future error, in order to discover high-error regions earlier, is an important area of future work.) Second, exploration uncovers novel parts of the space (e.g., the hood falling) that require further sampling to decrease maximum error back to previous levels.

We selected the **Robe** example because it contains large flowing regions that flop about with a relatively high degree of hysteresis. This behavior produces many secondary states per primary state, but nevertheless we end up with a reasonably sized compressed database. The **Casual** example is less hysteretic, and as a result requires less precomputation and produces a smaller compressed database.

5.2 Interactive Playback

We are able to play back cloth animation from the compressed secondary graph at well over 70 frames per second on an Apple MacBook Pro laptop (Core i7 CPU). Decompression as well as computation of the matrix-vector product to synthesize each cloth pose (which can be performed using fixed-point arithmetic) are performed using a simple CPU-based implementation parallelized using OpenMP.

6 Discussion

In this work we leveraged massive computing resources to push data-driven animation to unprecedented scales. While previous data-driven methods were limited to simple, controlled settings, we used thousands of hours of CPU-time to exhaustively explore the enormous phase space of secondary cloth motion. We believe the results of our large-scale exploration are striking. We not only capture a rich set of detailed cloth motions with sufficient density to eliminate playback artifacts for a wide range of character motions, we are able to discover bifurcations in the phase space and proceed to sample cloth dynamics in each of the distinct regions. The clothing in our demos does not simply return to a single rest state: hoods fall off and clothes wrap around the character in diverse ways.

A number of techniques were necessary to make large-scale exploration and subsequent real-time playback possible. Most importantly, we introduced the notion of a secondary motion graph to guide the phase-space exploration process and to enable efficient cloth response to run-time user input. Analysis of the reduced cloth basis led to a simple adaptive quantization scheme that further compressed the data set. We also created a cloth-pose similarity function that specifically accounted for the position of the robe's hood to robustly identify the most egregious secondary graph transition errors. All together, our system generated over 43 GB of cloth data, compressed this data into secondary cloth graphs of 34 MB and 66 MB in size, and delivered high-quality secondary cloth animation in response to user input on a laptop at over 70 fps.

A common concern about the viability of data-driven techniques focuses on run-time memory footprint. While our approximately 70 MB requirement is likely too large to be practical for games targeting modern console systems (for example, the Xbox 360 has only 512 MB of RAM), we believe its cost is modest in the context of today's modern PCs (and the coming generation of gaming consoles) which currently have multiple GBs of memory. Furthermore, we have not fully explored that gamut of cloth basis or secondary-graph compression strategies and so both better compression, as well as run-time solutions that stream regions of the secondary-graph (leaving only the basis representation in core), are likely possible.

We are excited by a number of potential benefits of the secondary motion graph representation. For example, the ability to view the entire database of cloth motion could be a significant advantage in the context of authoring a game. As part of the routine task of production game inspection, precomputed secondary cloth animation cloud be manually viewed, tagged for re-simulation, or even edited. Bad secondary graph transitions could be identified, adding a human component to what is currently an entirely procedural pose similarity metric. The result of this inspection effort would be high-quality, computationally-cheap, interactive clothing that is known to be free of objectionable artifacts.

More generally, while it is natural to consider graph exploration as a precomputation, we view phase-space sampling as an ongoing process whose improving results can be continually compressed and made available to an interactive system. Thus, phase-space exploration need not only proceed interactive playback as is typical of data-driven approaches, it may also be triggered and influenced as a result of it. Future on-line virtual worlds could deliver new secondary graphs to clients when new behaviors are discovered. Conversely, player actions in a virtual environment could indicate the most important areas of the phase space to sample. We believe there are many exciting possibilities for augmenting interactive environments with ongoing massive-scale computation.

Acknowledgements

The authors thank Alexander Reshetov and Alexei Soupikov for helping with code optimization, and Juan Miguel de Joya for modeling our character and garments. This work was supported by funding from the Intel Science and Technology Center for Visual Computing, NSF Grants IIS-0915462, IIS-0953985, and IIP-1127777, UC Lab Fees Research Program Grant 09-LR-01-118889-OBRJ, Samsung Scholarship (Woojong Koh), and gifts from Google, Qualcomm, Adobe, Pixar, the Okawa Foundation.

References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Transactions on Graphics 27*, 5 (Dec.), 165:1–165:10.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *Proc. of ACM SIGGRAPH '02*, 483–490.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of SIGGRAPH '98*, 43–54.

BARBIČ, J., AND JAMES, D. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics 24*, 3 (Aug.), 982–990.

BARBIČ, J., AND POPOVIĆ, J. 2008. Real-time control of physically based simulations using gentle forces. *ACM Transactions on Graphics 27*, 5 (Dec.), 163:1–163:10.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proc. of ACM SIGGRAPH '02*, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proc. '03 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 28–36.

CHOI, K.-J., AND KO, H.-S. 2002. Stable but responsive cloth. In *Proc. of ACM SIGGRAPH '02*, 604–611.

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

FENG, W.-W., YU, Y., AND KIM, B.-U. 2010. A deformation transformer for real-time cloth animation. *ACM Transactions on Graphics 1*, 212, 1–9.

GUAN, P., SIGAL, L., REZNITSKAYA, V., AND HODGINS, J. K. 2012. Multi-linear data-driven dynamic hair model with efficient hair-body collision handling. *Proc. '12 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 295–304.

HALEVY, A., NORVIG, P., AND PEREIRA, F. 2009. The unreasonable effectiveness of data. *IEEE Intelligent Systems 24*, 2 (Mar.), 8–12.

HILSMANN, A., AND EISERT, P. 2012. Image-based animation of clothes. *Eurographics*, 1–4.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. Tech. Rep. CMU-RI-TR-03-33, Carnegie Mellon University Robotics Institute.

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 (July), 582–585.

KALDOR, J. M., JAMES, D. L., AND MARSCHNER, S. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM Transactions on Graphics 29*, 4 (July), 1.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A. W., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4 (July).

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *Proc. of ACM SIGGRAPH '02*, 473–482.

MIGUEL, E., BRADLEY, D., THOMASZEWSKI, B., BICKEL, B., MATUSIK, W., OTADUY, M. A., AND MARSCHNER, S. 2012. Data-driven estimation of cloth simulation models. *Eurographics 31*, 2.

MÜLLER, M., AND CHENTANEZ, N. 2010. Wrinkle meshes. In *Proc. '10 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 85–92.

MÜLLER, M., RÖDER, T., CLAUSEN, M., EBERHARDT, B., KRÜGER, B., AND WEBER, A. 2007. Documentation mocap database HDM05. Tech. Rep. CG-2007-2, Universität Bonn, June.

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

NEALEN, A., MLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4, 809–836.

POPA, T., ZHOU, Q., BRADLEY, D., KRAEVOY, V., FU, H., SHEFFER, A., AND HEIDRICH, W. 2009. Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics 28*, 2.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation wrinkling : Augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Transactions on Graphics 29*, 6, 1–8.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proc. of ACM SIGGRAPH '87*, 205–214.

THOMASZEWSKI, B., WACKER, M., STRAER, W., LYARD, E., LUIBLE, C., VOLINO, P., KASAP, M., MUGGEO, V., AND MAGNENAT-THALMANN, N. 2007. Advanced topics in virtual garment simulation. In *Eurographics 2007 - Tutorials*, 795–855.

TREUILLE, A., LEWIS, A., AND POPOVIĆ, Z. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics 25*, 3 (July), 826–834.

TWIGG, C. D., AND JAMES, D. L. 2007. Many-worlds browsing for control of multibody dynamics. *ACM Transactions on Graphics 26*, 3 (July), 14:1–14:8.

TWIGG, C. D., AND JAMES, D. L. 2008. Backward steps in rigid body simulation. *ACM Transactions on Graphics 27*, 3 (Aug.), 25:1–25:10.

WANG, H., HECHT, F., RAMAMOORTHI, R., AND O'BRIEN, J. F. 2010. Example-based wrinkle synthesis for clothing animation. In *Proc. of ACM SIGGRAPH '10*, 107:1–8.

WANG, H., O'BRIEN, J. F., AND RAMAMOORTHI, R. 2011. Data-driven elastic models for cloth: modeling and measurement. *ACM Transactions on Graphics 30*, 4.

WICKE, M., STANTON, M., AND TREUILLE, A. 2009. Modular bases for fluid dynamics. *ACM Transactions on Graphics 28*, 3 (July), 39:1–39:8.