Precomputing Interactive Dynamic Deformable Scenes

Doug James Kayvon Fatahalian

CMU-RI-TR-03-33 2

Abstract

We present an approach for precomputing data-driven models of interactive physically based deformable scenes. The method permits real-time hardware synthesis of nonlinear deformation dynamics, including self-contact and global illumination effects, and supports real-time user interaction. We use data-driven tabulation of the system's deterministic state space dynamics, and model reduction to build efficient low-rank parameterizations of the deformed shapes. To support runtime interaction, we also tabulate *impulse response functions* for a palette of external excitations. Although our approach simulates particular systems under very particular interaction conditions, it has several advantages. First, parameterizing all possible scene deformations enables us to precompute novel reduced coparameterizations of global scene illumination for low-frequency lighting conditions. Second, because the deformation dynamics are precomputed and parameterized as a whole, collisions are resolved within the scene during precomputation so that runtime self-collision handling is implicit. Optionally, the data-driven models can be synthesized on programmable graphics hardware, leaving only the low-dimensional state space dynamics and appearance data models to be computed by the main CPU.

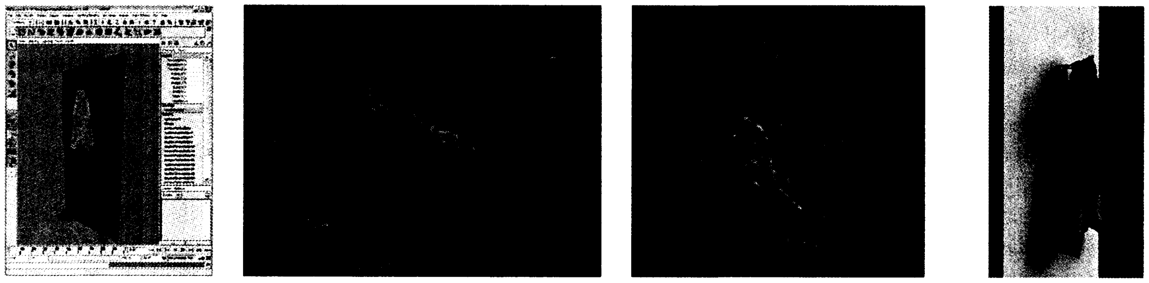
Contents

1 Introduction

Deformation is an integral part of our everyday world, and a key aspect of animating virtual creatures, clothing, fractured materials, surgical biomaterials, and realistic natural environments. It also constitutes a special challenge for real-time interactive environments. Designers of virtual environments may wish to incorporate numerous deformable components for increased realism, but often these simulations are only of secondary importance so very limited computing resources are available. Unfortunately, many realistic deformable systems are still notoriously expensive to simulate; robustly simulating large-scale nonlinear deformable systems with many self-collisions is fundamentally expensive [BFA02], and doing so with real-time constraints can be onerous. Perhaps as a consequence, very few (if any) major video games have appeared for which complex deformable physics is a substantial component. Numerous self-collisions complicate both runtime simulation and precomputation of interesting deformable scenes, and are also a hurdle for synthesizing physical models in real-time graphics hardware. Finally, realistic real-time animation of global illumination effects is also very expensive for deformable scenes, largely because it can not be precomputed as easily as for rigid models.

The goal of this paper is to strike a balance between complexity and interactivity by allowing certain types of interactive deformable scenes, with very particular user interactions, to be simulated at minimal runtime costs. Our method tabulates state space models of the system's deformation dynamics in a way that effectively allows interactive dynamics playback at runtime. To limit storage costs and increase efficiency, we project the state space models into very low-dimensional spaces using least-squares (Karhunen-Loève) approximations motivated by modal analysis. One might note that the highly complex geometry of dynamical systems' phase portraits, even for modest systems, suggests that it may be impractical to exhaustively sample the phase portrait [GH83, AS92]. Fortunately, this is unnecessary in our case. Our goal is not to exhaustively sample the dynamics to a specified accuracy, nor build a control system, instead we wish only to plausibly animate orbits from the phase portrait in a compelling interactive fashion. To this end, we sample the phase space dynamics using parameterized *impulse response functions* (IRFs) that have the benefit of being directly "playable" in a simulation provided the system is excited in similar contexts. We use small catalogues of interactions defined in discrete *impulse palettes* to constrain the range of user interaction, and thus reduce the effort required to tabulate system responses. This diminished range of interaction and control is a trade-off that can be suitable for virtual environments where interaction modalities are limited.

A major benefit of precomputing a reduced state space parameterization of deformable shapes is that we can also precompute a low-rank approximation to the scene's global illumination for real-time use. To address realistic appearance modeling we build on recent work by Sloan, Kautz and Snyder [SKS02] for radiance transfer approximations of global illumination for diffuse interreflections in low-frequency lighting environments. Data reduction is performed on the space of radiance transfer fields associated with the space of deformable models. The final low-rank deformation and transfer models can then be synthesized in real-time in programmable graphics hardware as in [JP02a, SKS02].



(a) Precomputation (b) Reduced dynamics model (c) Reduced illumination model (d) Real-time simulation

Figure 1: *Overview of our approach:* (a) Given a deformable scene, such as cloth on a user-movable door, we precompute (impulsive) dynamics by driving the scene with parameterized interactions representative of runtime usage. (b) Model reduction on observed dynamic deformations yields a low-rank approximation to the system's parameterized impulse response functions. (c) Deformed state geometries are then sampled and used to precompute and coparameterize a radiance transfer model for deformable objects. (d) The final simulation responds plausibly to interactions similar to those precomputed, includes complex collision and global illumination effects, and runs in real time.

1.1 Related Work

Deformable object simulation has had a long history in computer graphics, and enormous progress has been made [Wei86, TPBF87, PW89, BW98, OH99, BFA02]. More recently, attention has been given to techniques for interactive simulation, and several approaches have appeared that trade accuracy for real-time performance. For example, adaptive methods that exploit multiscale structure are very effective [DDCB01, GKS02, CGC$^+$02, JP03].

In this paper we are particularly interested in data-driven precompution for interactive simulation of deformable models. Prior work includes Green's function methods for linear elastostatics [CDA99, JP99, JP03], and modal analysis for linear elastodynamics [PW89, Sta97, JP02a]. These methods offer substantial speedups, but unfortunately do not easily generalize to more complex, nonlinear systems (although see [JP02b, KJP02] for quasistatic articulated models).

Dimensional model reduction using the ubiquitous principal component analysis (PCA) method [Nel00] is closely related to shape approximation methods used in graphics for compressing time-dependent geometry [Len99, AM00] and representing collections of shapes [BV99, SIC01]. For deformation, it is well-known that dimensional model reduction methods based on least-squares (Karhunen-Loève) expansions yield optimal modal descriptions for small vibrations [PW89, Sha90], and provide efficient low-rank approximations to nonlinear dynamical systems [Lum67]. We use similar least-squares model reduction techniques to reduce the dimension of our state space models. Finally, online data-reduction has been used to construct fast subspace projection (Newmark) time-stepping schemes [KLM01], however our goal is to avoid runtime time-stepping costs *entirely* by tabulating data-driven state space models using IRF primitives.

Our work is partly motivated by programmable hardware rendering of physical deformable models using low-rank linear superpositions of displacement fields. Applications include morphable models, linear modal vibration models [JP02a], and data-driven PCA mixture models for character skinning [KJP02]. Key differences are that (a) we address complex nonlinear dynamics with self-collisions, and (b) our appearances are based on low-rank approximations to radiance transfer fields instead of surface normal fields.

Data-driven tabulation of state space dynamics is an important strategy for identifying and learning how to control complex nonlinear systems [Nel00]. Grzeszczuk et al. [GTH98] trained neural networks to animate dynamical models with dozens of degrees of freedom, and learned the influence of several control parameters. Reissell and Pai [RP01] trained collections of autoregressive models with exogenous inputs (ARX models) to build interactive stochastic simulations of a candle flame silhouette and a falling leaf. In robotics, Atkeson et al. [AMS97] avoid the difficulties and effort of training a global regression model, such as neural networks or ARX models. Instead they use "locally weighted learning" to locally interpolate state space dynamics and control data, and only when needed at runtime, i.e., lazy learning. Our data-driven state space model differs from these three approaches in several ways. Most notably, our method sacrifices the quality of continuous control in favor of a simple discrete (impulsive) interaction. This allows us to avoid learning and (local) interpolation by using sampled data-driven IRF primitives that can be directly "played back" at runtime; this permits complex dynamics, such as nonsmooth contact and self-collisions, to be easily reproduced and avoids the need to generalize motions from possibly incomplete data. The simple IRF playback approach also avoids runtime costs associated with state model evaluation, e.g., interpolation. Another major difference is that our method uses model reduction to support very large dynamical systems with thousands or millions of degrees of freedom. Data-reduction quality can limit the effectiveness of the approach for large systems, but more sophisticated data compression techniques can be used. Finally, our state space model includes global illumination phenomena.

Our blending of precomputed orbital dynamics segments is related to Video Textures [SSSE00], wherein segments of video are rearranged and pieced together to form temporally coherent image sequences. This is also related to synthesis of character motions from motion capture databases using motion graphs [KGP02, LCR$^+$02]. Important differences are that our continuous deformable dynamics and appearance models can have a potentially much larger state space dimensionality, and the physical nature of data reduction is fundamentally different than, e.g., character motion. Also, the phenomena governing dynamic segment transitions are quite different, e.g., impulse responses can result in discontinuous transitions, and we are not concerned with the issue of control, so much as physical impulse resolution.

Global illumination and physically based deformation are historically treated separately in graphics. This is due in part to the fact that limited rendering precomputations can be performed, e.g., due to changing visibility. Consequently, real-time model animation has benefitted significantly from the advent of programmable graphics hardware [LJM01] for general lighting models [POAU00, PMTH01, OHHM02], stenciled shadows [EK02], ray tracing [PBMH02], interactive display of precomputed global illumination models [Hei01], and radiance transfer (for rigid models) [SKS02].

Our contribution: In this paper we introduce a precomputed data-driven state space modeling approach for generating real-time dynamic deformable models using black box offline simulators. This avoids the cost of traditional runtime computation of dynamic deformable models when not absolutely necessary. The approach is simple yet robust, can handle nonlinear deformations, self-contact, and large geometric models. The reduced phase space dynamics model also supports the precomputation and data reduction of complex radiance transfer global illumination models for real-time deformable scenes. Finally, the data-driven models allow dynamic deformable scenes to be compiled into shaders for (future) programmable graphics hardware.

Scope of Deformation Modeling: Our approach is broadly applicable to modeling deformable scenes, and can handle various complexities due to material and geometric nonlinearities, nonsmooth contact, and models of very large size. Given the combined necessity of (a) stationary statistics for model reduction and (b) sampling interactions for typical scenarios, the approach is most appropriate for scenes involving physical processes that do not undergo major irreversible changes, e.g., fracture. Put simply, the more repeatable a system's behavior is, the more likely a useful representation can be precomputed. Our examples involve structured, nonlinear, viscoelastic dynamic deformation; all models are attached to a rigid support, and reach equilibria in finite time (due to damping and collisions).

2 Data-Driven Deformation Modeling

At the heart of our data-driven simulator is a strategy for replaying appropriate precomputed impulse responses in response to user interactions. These dynamical time series segments, or *orbits* (after Poincaré [Poi57]), live in a high-dimensional *phase space*, and the set of all possible orbits composes the system's *phase portrait* [GH83]. In this section we first describe the basic compressed data-driven state space model.

2.1 Deterministic State Space Model

We model the discrete evolution of a system's combined dynamic deformation state, x, and globally illuminated appearance state, y, by an *autonomous deterministic state space model* [Nel00]:

$$\text{DYNAMICS}: \quad x^{(t+1)} = f(x^{(t)}, \alpha^{(t)}) \tag{1}$$
$$\text{APPEARANCE}: \quad y^{(t)} = g(x^{(t)}) \tag{2}$$

where at integer time step t,

- $x^{(t)}$ is the deformable *system state vector*, which describes the position and velocity of points in the deformable scene;
- $\alpha^{(t)}$ are *system parameters* describing various factors, such as generalized forces or modeled user interactions, that affect the state evolution from $x^{(t)}$ to $x^{(t+1)}$;
- $y^{(t)}$ are *dependent variables* defined in terms of the deformed state that describe our reduced appearance model but do not affect the deformation dynamics; and
- f and g are, in general, complicated nonsmooth functions that describe our dynamics and appearance models, respectively.

Different system models can have different definitions for x, α and y, and we will provide several examples later.

2.2 Data-driven State Spaces

Our data-driven deformation modeling approach involves tabulating the f function indirectly by observing time-stepped sequences of state transitions, $(x^{(t+1)}, x^{(t)}, \alpha^{(t)})$. By modeling deterministic *autonomous* state spaces, f does not explicitly depend on time, and precomputed tabulations can be reused later to simulate dynamics. Data-driven simulation involves carefully reusing these recorded state transitions to simulate the effect of $f(x, \alpha)$ for motions near the sampled state space.

Phase portrait notation: We model the state space as a collection of precomputed *orbits,* where each orbit is defined by a temporal sequence of *state nodes,* $x^{(t)}$, connected by *time step edges,* $e = (x^{(t+1)}, x^{(t)}, \alpha^{(t)})$. Without loss of generality, we can assume for simplicity that all time steps have a fixed step size Δt (which may be arbitrarily small). The collection of all precomputed orbits composes our *discrete phase portrait,* $\mathcal{P}$, and is a subset of the full phase portrait that describes all possible system dynamics. Our practical goal is to construct a $\mathcal{P}$ that provides a rich enough approximation to the full system for a particular range of interaction.

2.3 Dimensional Model Reduction

Discretizations of complex deformable models can easily involve thousands or millions of degrees of freedom (DOF). A cloth model with v moving vertices has $3v$ displacement and $3v$ velocity DOF, so the discrete phase portrait is composed of orbits evolving in $6v$ dimensions; for just 1000 vertices, data-driven state space modeling already requires tabulating dynamics in 6000 dimensions. Synthesizing large model dynamics directly, e.g., using state interpolation, would therefore be both computationally impractical and wasteful of memory resources.

To compactly represent the phase portrait $\mathcal{P}$, we first use model reduction to reduce state space dimensionality and exploit temporal redundancy. Here model reduction involves projecting the system's displacement (and other) field(s) into a low-rank basis derived from their observed dynamics. We note that the data reduction process is a black box step, but that we use the least-squares projection (PCA) since it provides an optimal description of small vibrations [Sha90], and can be effective for nonlinear dynamics [KLM01].

2.3.1 Model Reduction Details

Given the set of all N state nodes in $\mathcal{P}$ observed while time-stepping, we extract the vertex positions of each state's corresponding geometric mesh (for vertices we wish to later synthesize). By subtracting off the mean position of each vertex (or key representative shape), we obtain a displacement field for each state space node. Denote these N displacement fields as $\{\mathsf{u}^k\}_{k=1..N}$ (arbitrary ordering) where, for a model with v vertices, each $\mathsf{u}^k = (\mathsf{u}_i^k)_{i=1..v}$ has 3-vector components, and so is an M-vector with $M = 3v$. Let A_u denote the huge M-by-N dense displacement data matrix[1]

$$A_u = \left[\mathsf{u}^1 \mathsf{u}^2 \cdots \mathsf{u}^N\right] = \begin{bmatrix} \mathsf{u}_1^1 & \mathsf{u}_1^2 & & \mathsf{u}_1^N \\ \vdots & \vdots & \cdots & \vdots \\ \mathsf{u}_v^1 & \mathsf{u}_v^2 & & \mathsf{u}_v^N \end{bmatrix}. \tag{3}$$

Similar to linear elastodynamics where a small number of vibration modes can be sufficient to approximate observed dynamics, A_u can also be a low-rank matrix to a visual tolerance. We stably determine its low-rank structure by using a rank-r_u ($r_u \ll N$) Singular Value Decomposition (SVD) [GL96]

$$A_u \approx U_u S_u V_u^T \tag{4}$$

where U_u is an M-by-r_u orthonormal matrix with displacement basis vector columns, V_u is an N-by-r_u orthonormal matrix, and $S_u = \text{diag}(\sigma)$ is an r_u-by-r_u diagonal matrix with decaying singular values $\sigma = (\sigma_k)_{k=1...r_u}$, on the diagonal. The rank, r_u, of the approximation that guarantees a relative l_2 accuracy $\varepsilon_u \in (0, 1)$ is given by the largest r_u such that $\sigma_{r_u} \geq \varepsilon_u \sigma_1$ holds. Since A_u can be of gigabyte proportions we compute an approximate *output-sensitive SVD* with cost $\mathcal{O}(MNr_u)$ (see Appendix A).

2.3.2 Reduced State Vector Coordinates

The reduced displacement model induces a global reparameterization of the phase portrait, and yields the *reduced discrete phase portrait,* denoted by $\tilde{\mathcal{P}}$. The state vector is defined as

$$x = \begin{pmatrix} \mathsf{q}_u \\ \dot{\mathsf{q}}_u \end{pmatrix} \tag{5}$$

and its components are defined as follows.

[1]Let "u" ("a") subscripts denote displacement (appearance) data.

Reduced displacement coordinate, q_u: We define the r_u-by-N displacement coordinate matrix Q_u by

$$Q_u = S_u V_u^T = [\mathsf{q}_u^1 \mathsf{q}_u^2 \cdots \mathsf{q}_u^N] \tag{6}$$

such that

$$A_u \approx U_u Q_u \quad \Leftrightarrow \quad \mathsf{u}^k \approx U_u \mathsf{q}_u^k \tag{7}$$

where q_u^k is the *reduced displacement coordinate* of the k^{th} displacement field in the orthonormal displacement basis, U_u.

Reduced velocity coordinate, $\dot{\mathsf{q}}_u$: The *reduced velocity coordinate*, $\dot{\mathsf{q}}_u^k$, of the k^{th} state node could be defined similar to displacements, i.e., by reducing the matrix of all occurring velocity fields, however it is sufficient to define the reduced velocity using first-order finite differences. For example, we use a backward Euler approximation for each orbit (with forward Euler for the orbit's final state, and prior to IRF discontinuities),

$$\dot{\mathsf{u}}^k = (\mathsf{u}^{k+1} - \mathsf{u}^k)/\Delta t = (U_u \mathsf{q}_u^{k+1} - U_u \mathsf{q}_u^k)/\Delta t = U_u \dot{\mathsf{q}}_u^k. \tag{8}$$

Phase Portrait Distance Metric: Motivated by (7) and (8), a Euclidean metric for computing distances between two phase portrait states, x^1 and x^2, is given by

$$\mathrm{dist}(x^1, x^2) = \sqrt{\|\mathsf{q}_u^1 - \mathsf{q}_u^2\|_2^2 + \beta \|\dot{\mathsf{q}}_u^1 - \dot{\mathsf{q}}_u^2\|_2^2}, \tag{9}$$

where β is a parameter determining the relative (perceptual) importance of changes in position and velocity. Components of q_u and $\dot{\mathsf{q}}_u$ associated with large singular values can have dominant contributions. We choose β to balance the range of magnitudes of $\|\mathsf{q}\|_2$ and $\|\dot{\mathsf{q}}\|_2$ so that neither term overwhelms the other, and use

$$\beta = \max_{j=1..N} \|\mathsf{q}^j\|_2^2 / \max_{j=1..N} \|\dot{\mathsf{q}}^j\|_2^2 \tag{10}$$

as a default value.

3 Dynamics Precomputation Process

We precompute our models using offline simulation tools by crafting sequences of a small number of discrete impulses representative of runtime usage. Without the ability to resolve runtime user interactions, the runtime simulation would amount to little more than playing back motion clips.

3.1 Data-driven Modeling Complications

Several issues motivated our IRF simulation approach. Given the black box origin of the simulation, the function f is generally complex, and its interpolation is problematic for several reasons (see related Figure 2). Fundamental complications include insufficient data, high-dimensional state spaces, and divergence of nearby orbits. State interpolation also blurs important motion subtleties. Self-collisions result in complex configuration spaces that make generalizing tabulated motions difficult; an orbit tracking the boundary of an infeasible state domain, e.g., self-intersecting shapes, is surrounded by states that are physically invalid. For example, the cloth example will eventually, and very noticeably self-intersect if tabulated states are simply interpolated.

3.2 Impulse Response Functions

To balance these concerns and robustly support runtime interactions, we sample parameterized *impulse response functions (IRFs)* of the system throughout the phase portrait, and effectively replay them at run time. The key to our approach is that every sampled IRF is indexed by the initial state, x, and two user-modeled parameter vectors, α^I and α^F, that describe the initial *I*mpulse and persistent *F*orcing, respectively. In particular, given the system in state x, we apply a (possibly large) generalized force, parameterized by α^I, during the subsequent time step (See Figure 3). We then integrate the remaining dynamics for $(T - 1)$ steps with a generalized force that is parameterized by a value α^F

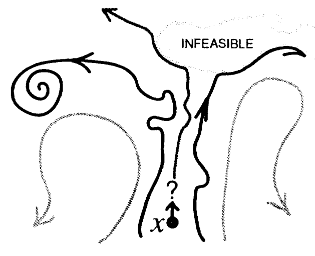


Figure 2: *Complications of data-driven dynamics:* Interpolating high-dimensional tabulated motions for a new state (starting at x) can be difficult in practice. One possible "realistic" orbit is drawn in red.

that remains constant throughout the remaining IRF orbit[2]. The tabulated IRF orbit is then a sequence of T time steps, $\left(e^1(\alpha^I), e^2(\alpha^F), \ldots, e^T(\alpha^F)\right)$, and we denote the IRF, ξ, by the corresponding sequence of states,

$$\mathsf{IRF}: \quad \xi(x, \alpha^I, \alpha^F; T) = \left(x^0 = x, x^1, \ldots, x^T\right), \tag{11}$$

with $\xi^t = x^t, t = 0, \ldots, T$.

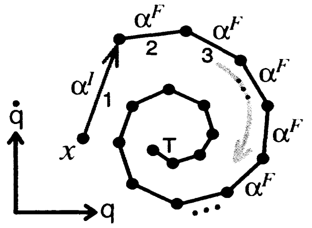


Figure 3: *The parameterized impulse response function (IRF), $\xi(x, \alpha^I, \alpha^F; T)$.*

An important special case of IRF occurs if the impulsive and persistent forcing parameters are identical, $\alpha^I = \alpha^F$. In this case, one α parameter describes each interaction type. See Figure 4 for a three parameter illustration, and Figure 5 for the cloth example.

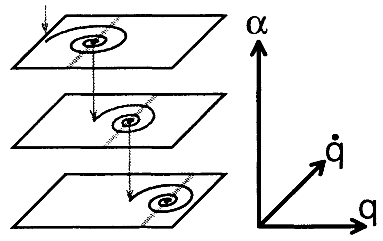


Figure 4: *A 3-valued $\alpha^I = \alpha^F$ system* showing orbits in $(\mathsf{q}_u, \dot{\mathsf{q}}_u)$-plane excited by changes between the three α values. The cloth-on-door example is analogous.

3.3 Impulse Palettes

To model the possible interactions during precomputation and runtime simulation, we construct an indexed *impulse palette* consisting of D possible IRF (α^I, α^F) pairs:

$$\mathcal{I}_D = \left((\alpha_1^I, \alpha_1^F), (\alpha_2^I, \alpha_2^F), \ldots, (\alpha_D^I, \alpha_D^F)\right). \tag{12}$$

Impulse palettes allow D specific interaction types to be modeled, and discretely indexed for efficient runtime access (described later in §5.2). For small D, the palette helps bound the precomputation required to sample the phase portrait to a sufficient accuracy. By limiting the range of possible user interactions, we can influence the statistical variability of the resulting displacement fields.

[2]Note: It follows from equation (1) that constant α does not imply constant forcing, since $x^{(t+1)}$ depends on both $\alpha^{(t)}$ *and* $x^{(t)}$.

3.4 Interaction Modeling

Impulse palette design requires some physical knowledge of the influence that discrete user interactions have on the system. For example, we now describe the three examples used in this paper (see accompanying Figure 5).

Dinosaur on moving car dashboard: The dinosaur model receives body impulse excitations resulting from discontinuous translational motions of its dashboard support. Our impulse palette models $D = 5$ pure impulses corresponding to 5 instantaneous car motions that shake the dinosaur followed by free motion: α_i^I describes the i^{th} body force 3-vector, and there are no persistent forces ($\alpha_i^F = 0$), i.e., $\mathcal{I}_D = \{(\alpha_1^I, 0), \dots, (\alpha_5^I, 0)\}$. (Since only translation is involved, gravitational force is constant in time, and does not affect IRF parameterization.)

Plant in moving pot: The pot is modeled as moving in a side-to-side motion with three possible speeds, $v \in \{-v_0, 0, +v_0\}$, and plant dynamics are analyzed in the pot's frame of reference. Since velocity dependent air damping forces are not modeled, the plant's equilibrium at speed $\pm v_0$ matches that of the pot at rest (speed 0). Therefore, similar to the dinosaur, we model the uniform body forcing as impulses associated with the left/right velocity discontinuities, followed by free motion (no persistent forcing) so that $\mathcal{I}_D = \{(-v_0, 0), (+v_0, 0)\}$.

Cloth on moving door: The door is modeled as moving at three possible angular velocities, $\omega \in \{-\omega_0, 0, +\omega_0\}$, with a 90 degree angular range. Air damping and angular acceleration induce a nonzero persistent force when $\omega = \pm\omega_0$, which is parameterized as $\alpha^F = \pm\omega_0$, and $\alpha^F = 0$ when $\omega = 0$. By symmetry, the cloth dynamics can be viewed in the frame of reference of the door, with velocity and velocity changes influencing cloth motion. Note that, unlike the translational motion of the plant's pot, the rotating door is not an inertial frame of reference. In this example no additional α^I impulse parameters are required, and we model the motion as the special case, $\alpha^I = \alpha^F$. Our impulse palette simply represents the three possible velocity forcing states $\mathcal{I}_D = \{(-\omega_0, -\omega_0), (0, 0), (\omega_0, \omega_0)\}$.

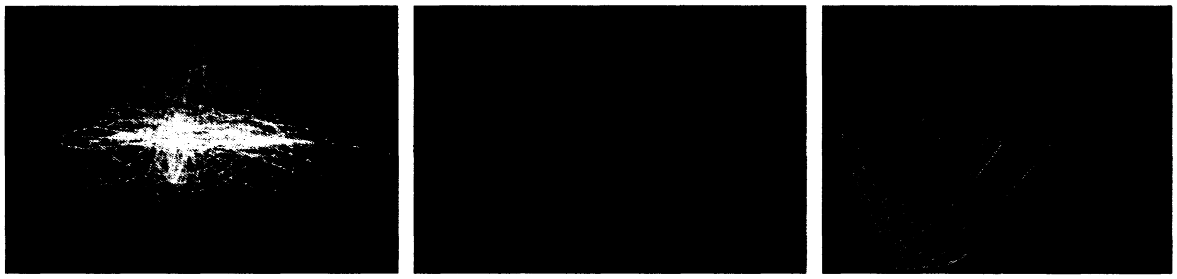


Figure 5: *Sampled IRF time-steps colored by impulse palette index (2D projection of* $(\mathsf{q}_u)_{1..3}$ *coordinates shown):* (Left) dinosaur with 5 impulse types; (Middle) plant with left (blue) and right (red) impulses; (Right) cloth with various door speeds: $-\omega_0$ (red), at rest (blue), and $+\omega_0$ (green).

3.5 Impulsively Sampling the Phase Portrait

By forcing the model with the impulse palette, IRFs can be impulsively sampled in the phase portrait. We use a simple random IRF sampling strategy pregenerated at the start of precomputation. A random sequence of impulse palette interactions are constructed, with each IRF integrated for a random duration, $T \in (T_{min}, T_{max})$, bounded by time scales of interest.

There are a couple of constraints on this process. First, the random sequence of impulse palette entries is chosen to be "representative of runtime usage" so that nonphysical or unused sequences do not occur. For example, the plant-in-pot example only models three pot motion speeds, $\{-v_0, 0, +v_0\}$, and therefore it is not useful to apply more than two same-signed $\pm v_0$ velocity discontinuities in a row. Similarly, the cloth's door only changes angular velocity by $|\omega_0|$ amounts, so that transitions from $-\omega_0$ to $+\omega_0$ are not sampled.

Second, a key point is that we sample enough IRFs of sufficiently long temporal duration to permit natural runtime usage. This is controlled using T_{min} and T_{max}. In particular, dynamics can either (a) be played to completion (if necessary), so that the model can naturally come to rest, or (b) expect to be interrupted based on physical modeling assumptions. As an example of the latter, the cloth's door has a 90 degree range of motion, so that IRFs associated with a nonzero angular velocity, ω, need be at most only $90/|\omega|$ seconds in duration.

In order to produce representative clips of motion, we filter sampled IRFs to discard the shortest ones. These short orbits are not a loss, since they are used to span the phase portrait. We also prune orbits that end too far away from the start of neighbouring IRFs, and are "dead ends." We can optionally extend sampled orbits during precomputation, e.g., if insufficient data exists. Orbits terminating close enough (in the phase space distance metric) to be smoothly blended to other orbits, e.g., using Hermite interpolation, can be extended. In more difficult cases where the dynamics are very irregular, such as for the cloth, one can resort to local interpolation, e.g., k-nearest neighbor, to extend orbits, but there are no quality guarantees. In general, we advocate sampling longer IRFs when possible.

While our sampling strategies are preplanned for use with standard offline solvers (see Figure 7), an online sampling strategy could be used. This would allow IRF durations, T, and new sampling locations to be determined at runtime, and could increase sampling quality.

4 Reduced Global Illumination Model

A significant benefit of precomputing parameterizations of the deformable scene is that complex data-driven appearance models can then also be precomputed for real-time use. This parameterized appearance model corresponds to the second part of our phase space model (Equation 2). Once the reduced dynamical system has been constructed, we precompute an appearance model based on a low-rank approximation to the diffuse radiance transfer global illumination model for low-frequency lighting [SKS02]. Unlike hard stenciled shadows, the diffuse low-frequency lighting model produces "blurry" lighting and is more amenable to statistical modeling of deformation effects.

4.1 Radiance Transfer for Low-frequency Lighting

Following [SKS02], for a given deformed geometry, for each vertex point, p, we compute the diffuse interreflected *transfer vector* $(M_p)_i$, whose inner product with the incident lighting vector, $(L_p)_i$, is the scalar exit radiance at p, or $L'_p = \sum_{i=1}^{n^2}(M_p)_i(L_p)_i$. Here both the transfer and lighting vectors are represented in spherical harmonic bases. For a given reduced displacement model shape[3], q_u, we compute the diffuse transfer field $\mathsf{M} = \mathsf{M}(\mathsf{q}_u) = (\mathsf{M}_{p_k})_{k=1..s}$ defined at s scene surface points, $p_k, k = 1..s$. Here M_{p_k} is a $3n^2$ vector for an order-n SH expansion and 3 color components, so that M is a large $3sn^2$ vector. We use $n = 4$ in all our examples, so that M has length $48s$, i.e., 48 floats per-vertex. Note that not all scene points are necessarily deformable, e.g., door, and some may belong to double-sided surfaces, e.g., cloth.

4.2 Dimensional Model Reduction

While we could laboriously precompute and store radiance transfer fields for all phase portrait state nodes, significant redundancy exists between them. Therefore, we also use least-squares dimensional model reduction to generate low-rank transfer field approximations. We note that, unlike displacement fields for which modal analysis suggests that least-squares projections can be optimal, there is no such motivation for radiance transfer.

Given N_a deformed scenes with deformation coordinates $(\mathsf{q}_u^1, \ldots, \mathsf{q}_u^{N_a})$, we compute corresponding scene transfer fields, $\mathsf{M}^j = \mathsf{M}(\mathsf{q}_u^j)$, and their mean, $\bar{\mathsf{M}}$. We substract the mean from each transfer field, $\tilde{\mathsf{M}}^j = \mathsf{M}^j - \bar{\mathsf{M}}$, and formally assemble them as columns of a huge $3sn^2$-by-N_a zero-mean[4] transfer data matrix,

$$A_a = \left[\tilde{\mathsf{M}}^1 \tilde{\mathsf{M}}^2 \cdots \tilde{\mathsf{M}}^{N_a}\right] = \begin{bmatrix} \tilde{\mathsf{M}}_{p_1}^1 & \tilde{\mathsf{M}}_{p_1}^2 & \cdots & \tilde{\mathsf{M}}_{p_1}^{N_a} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{\mathsf{M}}_{p_s}^1 & \tilde{\mathsf{M}}_{p_s}^2 & \cdots & \tilde{\mathsf{M}}_{p_s}^{N_a} \end{bmatrix}. \tag{13}$$

We compute the SVD of A_a to determine the reduced low-rank transfer model, and so discover the reduced transfer field coordinates $\mathsf{q}_a^j = \mathsf{q}_a(\mathsf{q}_u^j), j = 1 \ldots N_a$, for the N_a deformed scenes. We denote the final rank-r_a factorization

[3]The appearance model depends on the deformed shape of the scene (q_u) but not its velocity ($\dot{\mathsf{q}}_u$).

[4]Formally, there is no need to subtract the data mean prior to SVD (unlike for PCA where the covariance matrix must be constructed), but we do so because the first coordinate captures negligible variability otherwise.

as $A_a \approx U_a Q_a$ where U_a are the orthonormal transfer field basis vectors, and $Q_a = [\mathsf{q}_a^1 \cdots \mathsf{q}_a^{N_a}]$ are the reduced appearance coordinates.

4.3 Interpolating Sparsely Sampled Appearances

Computing radiance transfer for all state nodes can be very costly and also perceptually unnecessary. We therefore interpolate the reduced radiance transfer fields across the phase portrait. Normalized Radial Basis Functions (NRBFs) are a natural choice for interpolating high-dimensional scattered data [Nel00]. We use K-means clustering [Nel00] to cluster phase portrait states into $N_a \ll N$ clusters (see Figure 6). A representative state q_u^k closest to the k^{th} cluster's mean is used to compute radiance transfer (using the original state node's unreduced mesh to avoid compression artifacts in the lighting model). Model reduction is then performed on the radiance transfer fields for the N_a states. In the end, we know the reduced radiance values q_a^k at N_a state nodes, i.e., $\mathsf{q}_a^k = \mathsf{q}_a(\mathsf{q}_u^k), \quad k = 1 \ldots N_a$. These sparse samples are then interpolated using a regularized NRBF approach (see Appendix B). This completes the definition of the deterministic state space model originally referred to in Equation 2.

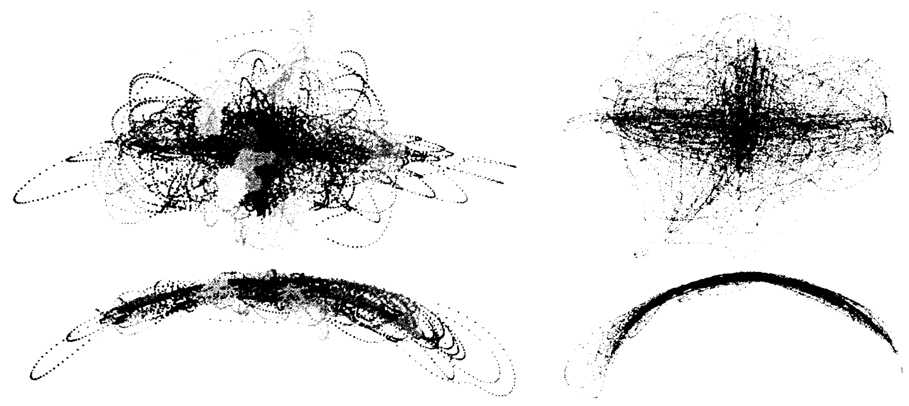


Figure 6: *Clustering of deformed dinosaur scenes for transfer computation:* (Left) Clustered shape coordinates $\{\mathsf{q}_u\}$; (Right) interpolated appearance coordinates $\{\mathsf{q}_a\}$. Only the first three (2D-projected) components of q are displayed.

5 Runtime Synthesis

At any runtime instant, we either "play an IRF" or are at rest, e.g., at the end of an IRF. Once an impulse is specified by an index from the impulse palette, we switch to a nearby IRF of that type and continue until either interrupted by another impulse signal or we reach the end of the IRF and come to rest. At each time-step we trivially lookup/compute q_u and q_a and use them to evaluate matrix-vector products for the displacement $\mathsf{u} = U_u \mathsf{q}_u$ and radiance transfer $\mathsf{M} = U_a \mathsf{q}_a$ fields needed for graphical rendering. This approach has the benefits of being both simple and robust.

5.1 Approximating Impulse Responses

Given the problems associated with orbit interpolation (§3.1), we wish to transition directly between IRFs during simulation. To avoid transition (popping) artifacts, we smoothly transition between the state and the new IRF. We approximate the IRF at x' with the IRF $\xi^t(x, \alpha^I, \alpha^F)$ from a nearby state x by adding an exponentially decaying state offset $(x' - x)$ to the state,

$$\xi^t(x', \alpha^I, \alpha^F; T) \approx \xi^t(x, \alpha^I, \alpha^F; T) + (x' - x)e^{-\lambda t}, \tag{14}$$

where $t = 0, 1, \ldots, T$, and $\lambda > 0$ determines the duration of the offset decay. This approximation converges as $x' \to x$, e.g., as $\mathcal{P}$ is sampled more densely, but its chief benefit is that it can produce plausible motion even in undersampled cases (as in Figure 2). For rendering, appearance coordinates associated with $\xi^t(x, \alpha^I, \alpha^F; T)$ are also averaged,

$$\mathsf{q}_a^t(x', \alpha^I, \alpha^F; T) \approx \mathsf{q}_a^t(x, \alpha^I, \alpha^F; T) + (\mathsf{q}_a(x') - \mathsf{q}_a(x))e^{-\lambda t}. \tag{15}$$

Finally, the cost of computing the approximate shape and appearance coordinates is proportional to the dimension of the reduced coordinate vectors, $2r_u + r_a$, and is cheap in practice.

5.2 Caching Approximate IRF References

A benefit of using a finite impulse palette $\mathcal{I}_D$ is that each state in the phase portrait can cache the D references to the nearest corresponding IRFs. Then at runtime, for the system in (or near) a given phase portrait state, x, the response to any of the D impulse palette excitations can be easily resolved using table lookup and IRF blending. By caching these local references at state nodes it is in principle possible to verify during precomputation that blending approximations are reasonable, and, e.g., don't lead to self-intersections.

5.3 Low-rank Model Evaluation

The two matrix-vector products, $\mathsf{u} = U_u \mathsf{q}_u$ and $\tilde{\mathsf{M}} = U_a \mathsf{q}_a$, can be evaluated in software or in hardware. Our hardware implementation is similar to [JP02a, SKS02] in that we compute the per-vertex linear superposition of displacements and transfer data in vertex programs. Given the current per-vertex attribute memory limitations of vertex programs (64 floats), some superpositions must be computed in software for larger ranks. Similar to [SKS02], we can reduce transfer data requirements (by a factor of $n^2 = 16$) by computing and caching the $3r_a$ per-vertex light vector inner-products in software, i.e., fixing the light vector. Each vertex's color linear superposition then involves only r_a 3-vectors.

Scene	Deformable Model					Appearance Model						IRF	
	F	V	DOF	r_u	relErr	F	V_{lit}	DOF	N_a	r_a	relErr	N	#IRF
Dinosaur	49376	24690	74070	12	0.5%	52742	26361	1265328	50	7	4.4%	20010	260
Cloth	6365	3310	9930	30	1.5%	25742	16570	795360	200	12	15%	8001	171
Plant	6304	3750	11250	18	2.0%	11184	9990	479520	100	12	6.2%	6245	150

Table 1: *Model Statistics:* The r_u and r_a ranks correspond to those used for frame rate timings in Table 2.

6 Results

We applied our method to three deformable scenes typical of computer graphics that demonstrate some strengths and weaknesses of our approach. Dynamics and transfer precomputation and rendering times are given in Table 2, and model statistics are in Table 1. The dynamics of each of the three models were precomputed over the period of a day or more using standard software (see Figure 7). Radiance transfer computations took on the order of a day or more due to polygon counts, and sampling densities, N_a.

Scene	Precomputation		Frame rates (in SW)	
	Dynamics	Transfer	Defo	Defo + Trnsfr
Dinosaur	33h 21m	74h	173	82 (175 in HW)
Cloth	16h 46m	71h 27m	350	149
Plant	$\approx$ 1 week	11h 40m	472	200

Table 2: *Model timings* on an Intel Xeon 2.0GHz, 2GB-266MHz DDR RDRAM, with GeForce FX 5800 Ultra. Runtime matrix-vector multiplies computed in software (SW) using an Intel performance library, except for the dinosaur example that fits into vertex program hardware (HW).

The cloth example demonstrates interesting soft shadows and illumination effects, as well as subtle nonlinear deformations typical of cloth attached to a door (see Figure 8). This is a challenging model for data-driven deformation because the material is thin, and in persistent self-contact, so that small deformation errors can result in noticeable self-intersection. By increasing the rank of the deformation model (to $r_u = 30$), the real-time simulation avoids visible intersection[5]. Unlike the other examples, the complex appearance model appears to have been undersampled by

[5]Using a larger cloth "thickness" during precomputation would also reduce intersection artifacts.

Figure 7: *Precomputing real-time models with offline simulators:* (Left) cloth precomputed in Alias|Wavefront Maya; (Middle,Right) models precomputed by an engineering analysis package (ABAQUS) using an implicit Newmark integrator.

the cluster shape sampling ($N_a = 200$) since there is some uneveness in interpolated lighting and certain cloth-door shadows lack proper variation.

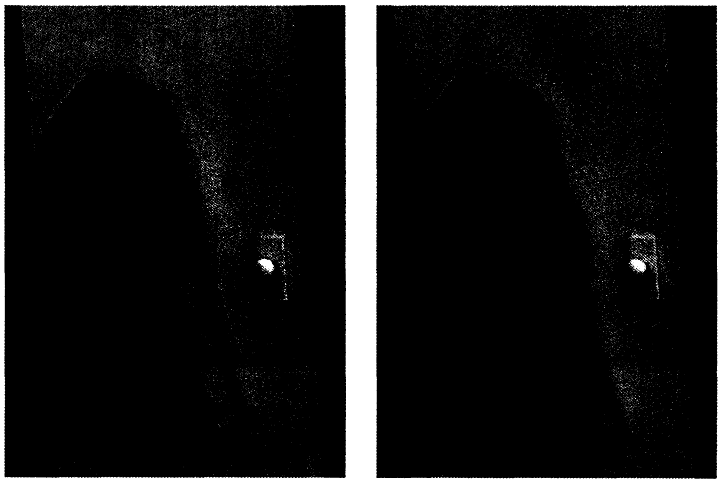
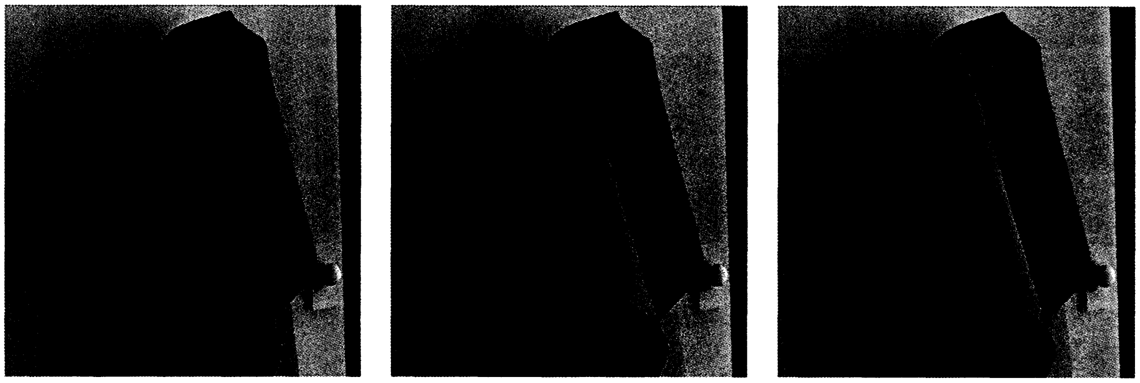


Figure 8: *Dynamic cloth states* induced by door motions: (Left) cloth and door at rest; (Right) cloth pushed against moving door by air drag.

The plant example has interesting shadows and changing visibility, and the dynamics involve significant multi-leaf collisions (see Figures 10, 11 and 12). The plant was modeled with 788 quadrilateral shell finite elements in ABAQUS, with the many collisions accurately resolved with barrier forces and an implicit Newmark integrator.

The rubber dinosaur had the simplest dynamics of the three models, and did not involve collisions. It was precomputed using an implicit Newmark integrator with 6499 FEM tetrahedral elements, and displacements were interpolated onto a finer displaced subdivision surface mesh [LMH00]. However, the radiance transfer data model was easily the largest of the three, and includes interesting self-shadowing (on the dinosaur's spines) and color bleeding (see Figure 11). Runtime simulation images, and some reduced basis vectors, or modes, of the dinosaur's displacement and radiance transfer models are shown in Figure 14.

All examples display interesting dynamics and global illumination behaviors. Significant reductions in the rank of the displacement and appearance models were observed, with only a modest number of flops per vertex required to synthesize each deformed and illuminated shape. In particular, Figure 13 shows that the singular values converge quickly, so that useful approximations are possible at low ranks. In general, for a given error tolerance, appearance models are more complex than deformation models. However, when collisions are present, low-rank deformation approximations can result in visible intersection artifacts so that somewhat higher accuracy is needed.



$r_a = 0$ (mean) $r_a = 3$ $r_a = 6$

Figure 9: *Reduced radiance transfer illuminations* for an arbitrary cloth pose illustrate improving approximations as the rank r_a is increased.

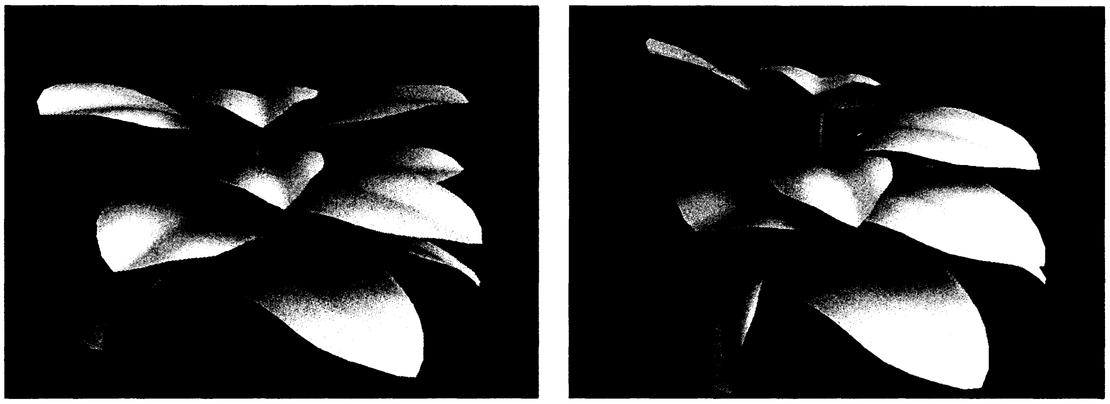


Figure 10: *View of plant from behind*

7 Summary and Discussion

Our method permits interactive models of physically based deformable scenes to be almost entirely precomputed using data-driven tabulation of state space models for shape and appearance. Using efficient low-rank approximations for the deformation and appearance models, extremely fast rendering rates can be achieved for interesting models. Such approaches can assist traditional simulation in applications with constrained computing resources.

Future work includes improving dynamics data quality by using better non-random sampling strategies for IRFs. Building better representations for temporal and spatial data could be investigated using multiresolution and "best basis" statistics. Generalizing the possible interactions, e.g., for contact, and supporting more complex scenes remain important problems. Extensions to deformable radiance transfer for nondiffuse models, and importance-based sparse shape sampling are also interesting research areas.

Acknowledgments: We would like to acknowledge the helpful criticisms of several anonymous reviewers, assistance from Christopher Twigg and Matthew Trentacoste, hardware from NVIDIA, Cyberware for the dinosaur model, and the Carnegie Mellon Graphics group.

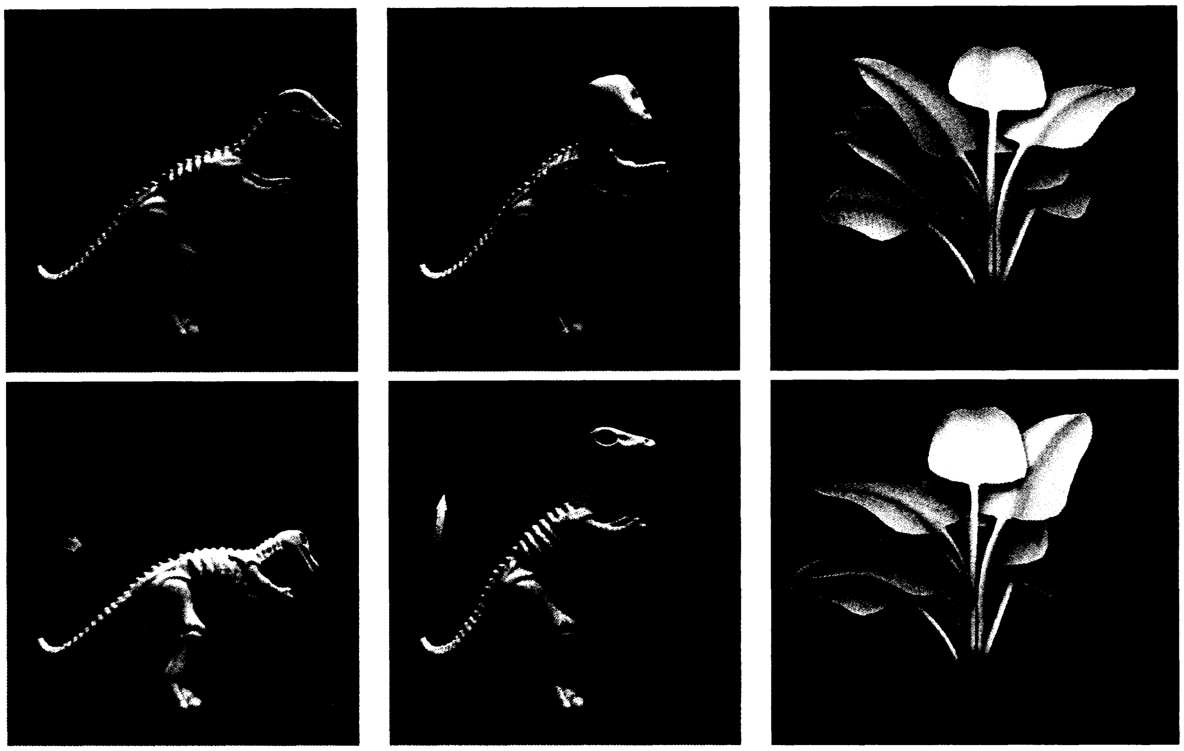
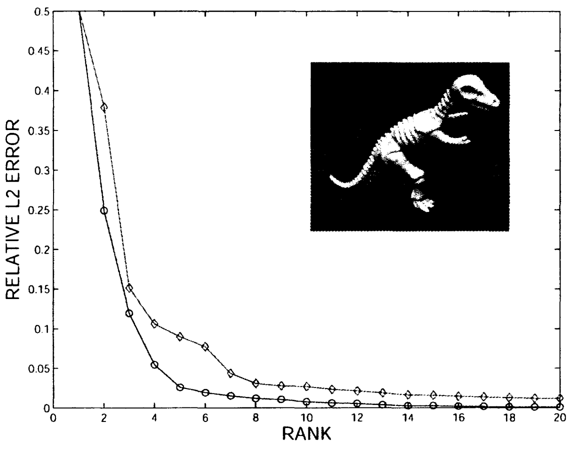


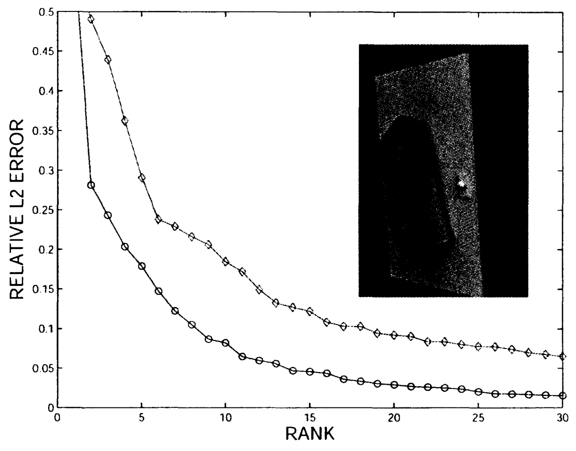
Figure 11: *Interactive dynamic behaviors resulting from applied impulses*

Figure 12: *Dynamic interpolated shadows*

A Output-Sensitive SVD

Computing the Singular Value Decomposition (SVD) of a huge m-by-n matrix is very costly ($\mathcal{O}(mn\min(m,n))$ [GL96]), and is wasteful for low-rank matrices, especially when only leading singular factors are desired. Fortunately fast out-of-core algorithms exist, and we use the following simple direct method to approximate the largest orthogonal singular factors for a large approximately low-rank m-by-n matrix, A. We begin by computing a drop tolerance rank-r unmodified Graham-Schmidt orthogonal factorization, $A \approx EX$, for a given sufficiently small tolerance $\varepsilon \in (0,1)$, where E is an m-by-r orthogonal Graham basis matrix, and X is an r-by-n matrix of expansion coefficients. The factorization proceeds by processing one column of A at a time, so that the matrix A is never explicitly constructed. For a given column, $A_{:j}$, and current rank of the Graham basis, $\tilde{r}$, the basis is expanded only if the relative residual





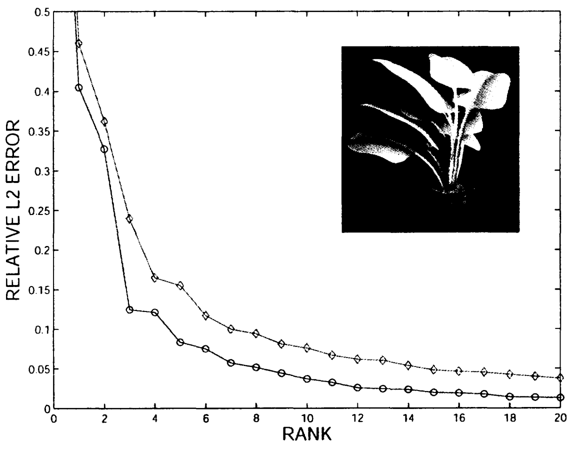


Figure 13: *Model reduction:* Relative l_2 error versus model rank, r_u and r_a, for the displacement (blue) and radiance transfer (red) models.

error is greater than the (drop tolerance) threshold, ε, i.e.,

$$\|A_{:j} - \tilde{A}_{:j}\| > \varepsilon \|A_{:j}\|, \tag{16}$$

where $\tilde{A}_{:j}$ is the projection of $A_{:j}$ into the current rank-$\tilde{r}$ Graham basis. The final low-rank SVD is then found by taking the SVD of $X = \tilde{U}SV^T$ and multiplying as follows:

$$A \approx EX = E(\tilde{U}SV^T) = (E\tilde{U})SV^T = USV^T. \tag{17}$$

In practice we discard all factors whose singular values when normalized by the largest value are less than some desired accuracy ($> \varepsilon$). We refer the interested reader to [GL96, Bra02, SZ00] for related algorithms.

B NRBF Appearance Interpolation

Normalized radial basis functions (NRBFs) are used to interpolate $\mathsf{q}_a(\mathsf{q}_u)$ at all phase portrait q_u. We center radial basis functions at all of the M_a transfer sampled state nodes,

$$\phi_k(\mathsf{q}_u) = \exp(\|\mathsf{q}_u^k - \mathsf{q}_u\|_2^2/a^2), \quad k = 1 \ldots M_a, \tag{18}$$

where a is the radial scale of the basis functions. The NRBF interpolant is

$$\mathsf{q}_a^{RBF}(\mathsf{q}_u) = \sum_k \hat{\phi}_k(\mathsf{q}_u)\mathsf{c}_a^k \tag{19}$$

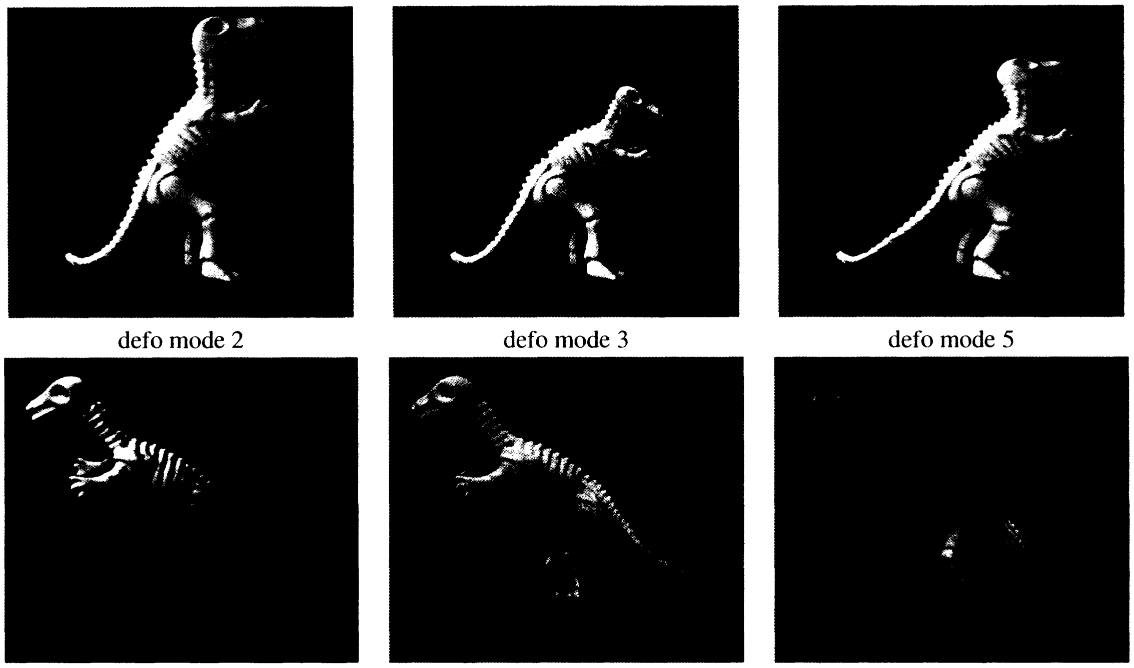


Figure 14: *Dinosaur modes:* (Top) displacement mode shapes; (Bottom) radiance transfer modes.

where c_a^k are the unknown coefficients, and $\hat{\phi}_k(\mathsf{q}_u)$ are normalized basis functions,

$$\hat{\phi}_k(\mathsf{q}_u) = \frac{\phi_k(\mathsf{q}_u)}{\sum_i \phi_i(\mathsf{q}_u)}. \tag{20}$$

The components of c_a^k can be determined by substituting the data, $\mathsf{q}_a^k = \mathsf{q}_a(\mathsf{q}_u^k), \quad k = 1 \ldots M_a$, to obtain

$$\mathsf{q}_a^{RBF}(\mathsf{q}_u^i) = \sum_j \hat{\phi}_j(\mathsf{q}_u^i)\mathsf{c}_a^j = \sum_j B_{ij}\mathsf{c}_a^j, \quad i = 1 \ldots M_a. \tag{21}$$

To avoid ill-conditioning which occurs as M_a or a increase, i.e., over-fitting, the NRBF equation is solved using truncated SVD.

References

[AM00] Marc Alexa and Wolfgang Müller. Representing Animations by Principal Components. *Computer Graphics Forum*, 19(3):411–418, August 2000.

[AMS97] C.G. Atkeson, A.W. Moore, and S. Schaal. Locally Weighted Learning for Control. *AI Review*, 11:75–113, 1997.

[AS92] Ralph H. Abraham and Christopher D. Shaw. *Dynamics - the Geometry of Behavior.* Addison-Wesley, 1992.

[BFA02] Robert Bridson, Ronald P. Fedkiw, and John Anderson. Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. *ACM Transactions on Graphics*, 21(3):594–603, July 2002.

[Bra02] Matthew E. Brand. Incremental Singular Value Decomposition of Uncertain Data with Missing Values. In *Proc. of Euro. Conf. on Comp. Vision (ECCV)*, volume 2350, pages 707–720, May 2002.

[BV99] Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Proc. of SIGGRAPH 99*, pages 187–194, August 1999.

[BW98] David Baraff and Andrew P. Witkin. Large Steps in Cloth Simulation. In *Proceedings of SIGGRAPH 98*, pages 43–54, July 1998.

[CDA99] S. Cotin, H. Delingette, and N. Ayache. Realtime Elastic Deformations of Soft Tissues for Surgery Simulation. *IEEE Trans. on Vis. and Comp. Graphics*, 5(1):62–73, 1999.

[CGC$^+$02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A Multiresolution Framework for Dynamic Deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 41–48, July 2002.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. In *Proceedings of SIGGRAPH 2001*, pages 31–36, August 2001.

[EK02] Cass Everitt and Mark J. Kilgard. Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering. Technical report, NVIDIA Corporation, Inc., Austin, Texas, 2002.

[GH83] John Guckenheimer and Philip Holmes. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields (Appl. math. sci.; v. 42).* Springer-Verlag New York, Inc., 1983.

[GKS02] Eitan Grinspun, Petr Krysl, and Peter Schröder. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Transactions on Graphics*, 21(3):281–290, July 2002.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, third edition, 1996.

[GTH98] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton. NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models. In *Proceedings of SIGGRAPH 98*, pages 9–20, July 1998.

[Hei01] Wolfgang Heidrich. Interactive Display of Global Illumination Solutions for Non-diffuse Environments - A Survey. *Computer Graphics Forum*, 20(4):225–244, 2001.

[JP99] Doug L. James and Dinesh K. Pai. ARTDEFO - Accurate Real Time Deformable Objects. In *Proc. of SIGGRAPH 99*, pages 65–72, August 1999.

[JP02a] Doug L. James and Dinesh K. Pai. DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Trans. on Graphics*, 21(3):582–585, July 2002.

[JP02b] Doug L. James and Dinesh K. Pai. Real Time Simulation of Multizone Elastokinematic Models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 927–932, Washington, DC, 2002.

[JP03] Doug L. James and Dinesh K. Pai. Multiresolution Green's Function Methods for Interactive Simulation of Large-scale Elastostatic Objects. *ACM Trans. on Graphics*, 22(1):47–82, 2003.

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion Graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[KJP02] Paul G. Kry, Doug L. James, and Dinesh K. Pai. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. In *SIGGRAPH Symposium on Computer Animation*, pages 153–160, July 2002.

[KLM01] P. Krysl, S. Lall, and J. E. Marsden. Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering*, 51:479–504, 2001.

[LCR$^+$02] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive Control of Avatars Animated With Human Motion Data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[Len99] Jerome Edward Lengyel. Compression of Time-Dependent Geometry. In *ACM Symposium on Interactive 3D Graphics*, pages 89–96, April 1999.

[LJM01] Erik Lindholm, Mark J.Kilgard, and Henry Moreton. A User-Programmable Vertex Engine. In *Proceedings of SIGGRAPH 2001*, pages 149–158, Aug 2001.

[LMH00] A. Lee, H. Moreton, and H. Hoppe. Displaced Subdivision Surfaces. In *Proc. of SIGGRAPH 2000*, pages 85–94, 2000.

[Lum67] J. L. Lumley. The structure of inhomogeneous turbulence. In *Atmospheric turbulence and wave propagation*, pages 166–178, 1967.

[Nel00] Oliver Nelles. *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer Verlag, December 2000.

[OH99] J.F. O'Brien and J.K. Hodgins. Graphical Modeling and Animation of Brittle Fracture. In *SIGGRAPH 99 Conference Proceedings*, pages 111–120, 1999.

[OHHM02] Marc Olano, John C. Hart, Wolfgang Heidrich, and Michael McCool. *Real-Time Shading*. A.K.Peters, 2002.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray Tracing on Programmable Graphics Hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002.

[PMTH01] Kekoa Proudfoot, William R. Mark, Svetoslav Tzvetkov, and Pat Hanrahan. A Real-Time Procedural Shading System for Programmable Graphics Hardware. In *Proceedings of SIGGRAPH 2001*, pages 159–170, 2001.

[POAU00] Mark S. Peercy, Marc Olano, John Airey, and P. Jeffrey Ungar. Interactive Multi-Pass Programmable Shading. In *Proceedings of SIGGRAPH 2000*, pages 425–432, 2000.

[Poi57] H. Poincaré. *Les Méthodes Nouvelles de la Mécanique Célèste I, II, III.* (Reprint by) Dover Publications, 1957.

[PW89] A. Pentland and J. Williams. Good Vibrations: Modal Dynamics for Graphics and Animation. In *Computer Graphics (SIGGRAPH 89)*, volume 23, pages 215–222, 1989.

[RP01] L. M. Reissell and Dinesh K. Pai. Modeling Stochastic Dynamical Systems for Interactive Simulation. *Computer Graphics Forum*, 20(3):339–348, 2001.

[Sha90] A.A. Shabana. *Theory of Vibration, Volume II: Discrete and Continuous Systems.* Springer–Verlag, New York, NY, first edition, 1990.

[SIC01] Peter-Pike J. Sloan, Charles F. Rose III, and Michael F. Cohen. Shape by Example. In *ACM Symp. on Interactive 3D Graphics*, pages 135–144, March 2001.

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002.

[SSSE00] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video Textures. In *Proc. of SIGGRAPH 2000*, pages 489–498, July 2000.

[Sta97] J. Stam. Stochastic Dynamics: Simulating the Effects of Turbulence on Flexible Structures. *Computer Graphics Forum*, 16(3), 1997.

[SZ00] Horst D. Simon and Hongyuan Zha. Low-Rank Matrix Approximation Using the Lanczos Bidiagonalization Process with Applications. *SIAM Journal on Scientific Computing*, 21(6):2257–2274, 2000.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically Deformable Models. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, volume 21(4), pages 205–214, July 1987.

[Wei86] Jerry Weil. The Synthesis of Cloth Objects. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20(4), pages 49–54, August 1986.