Inverse-Foley Animation: Synchronizing rigid-body motions to sound

Timothy R. Langlois Doug L. James

Cornell University

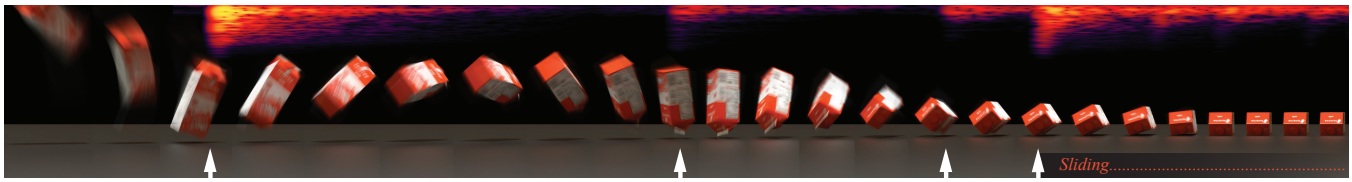


Figure 1: Rigid-body motion from sound: *Given a recording of contact sounds produced by passive object dynamics, Inverse Foley Animation can estimate plausible rigid-body motions that have similar contact events occuring at similar times (*Carton *example).*

Abstract

In this paper, we introduce *Inverse-Foley Animation*, a technique for optimizing rigid-body animations so that contact events are synchronized with input sound events. A precomputed database of randomly sampled rigid-body contact events is used to build a contact-event graph, which can be searched to determine a plausible sequence of contact events synchronized with the input sound's events. To more easily find motions with matching contact times, we allow transitions between simulated contact events using a motion blending formulation based on modified contact impulses. We fine tune synchronization by slightly retiming ballistic motions. Given a sound, our system can synthesize synchronized motions using graphs built with hundreds of thousands of precomputed motions, and millions of contact events. Our system is easy to use, and has been used to plan motions for hundreds of sounds, and dozens of rigid-body models.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation; H.5.5 [Information Systems]: Information Interfaces and Presentation—Sound and Music Computing;

Keywords: motion control, rigid-body dynamics, motion graphs, sound design, sound rendering, sound synthesis

Links: DL PDF WEB

1 Introduction

> *Synchresis: "... the spontaneous and irresistible mental fusion, completely free of any logic, that happens between a sound and a visual when these occur at exactly the same time."* [Chion 1994]

If you hear contact sounds of an object bouncing around on the floor, you can probably create a plausible picture in your mind of what is happening. Unfortunately, computers and robots do not know how to hallucinate such motions just from the sound alone. In this work, we explore how contact sounds can be used to automatically synthesize plausible synchronized animations. Beyond pure intellectual curiosity, there are several reasons for doing so. Applications include low-cost sound-based motion capture, where plausible motion can be estimated or designed using sound alone. Sound can also help improve video-based estimation of rigid-body motion by providing contact event times, especially for blurry images of fast moving or small objects. In computer animation and games, synchronized contact sounds are commonly added "after the fact," by hand, or with event-triggered sound clips, or using digital sound synthesis. Unfortunately, manually adding sound to complex animated phenomena can degrade audiovisual synchronization, especially for rigid body motion, where mis-synchronized sound and impact events can be apparent. In contrast, digital sound synthesis can compute synchronized sound directly from the physics-based computer animation, but it can be difficult to achieve the realism of recorded sounds for many common objects. In this paper, we explore an alternate computational way to ensure audio-visual synchronization for physically based animation, while retaining the richness of pre-recorded and custom-designed sound effects.

Our technique, *Inverse-Foley Animation* (IFA), optimizes rigid-body animations to synchronize contact events with input sound events. We explore this sound-constrained motion design problem in the context of passive rigid-body dynamics (see Figure 1).

Given an input recording of contact sounds produced by a single object, a user indentifies contact event times when simulated rigid-body contact events likely occur. A virtual rigid-body dynamics model is constructed that approximates the scenario in which the sound was recorded. Since it is very difficult, if not impossible, to optimize a rigid-body simulation's parameters (initial conditions, restitution, friction, geometry, etc.) to produce contact events at the exact same times, we use the following data-driven approach. First, stochastic sampling is used to generate a rigid-body motion database, which we interpret as a library of rigid-body contact events (nodes) connected by ballistic rigid-body motions (edges). To help synthesize plausible contact event sequences that match the desired event times, we construct a contact-event graph, and introduce additional transition edges between contact-event nodes with similar attributes, e.g., similar orientation and velocity. By precomputing motion on a plane, we are able to globally adjust position, and planar orientation of contact events, using suitable transformations. We blend from one post-impact trajectory to another by optimizing a small impulse correction to match the subsequent contact state, e.g., orientation. A cost function is used to penalize implausible motion transitions, and to better synchronize with the sound. Modest motion retiming is used to obtain perfect synchronization

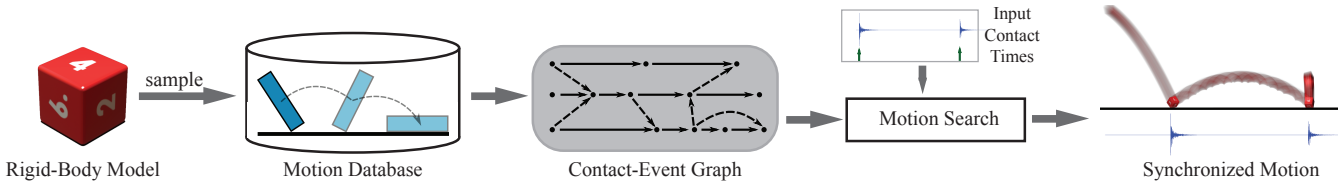


Figure 2: Overview

with the input sound. Our contact-event graph can be searched for plausible motion paths, and supports constraints so the final contact event occurs at a contact-event node which is a terminal resting state. Additional contact constraints can be also be introduced, such as to make an object land in a particular orientation, or to match contact locations observed in a video capture (see Figure 14). An overview of our approach is shown in Figure 2.

Using our approach we were able to generate plausible rigid-body animations with realistic synchronized sound. Our system has successfully synthesized motions for dozens of objects (see Figure 13) and hundreds of sounds, many of which would be hard to synthesize sounds for digitally, e.g., a scruffy bulb of garlic.

Our technique also provides a new way for animators to use sound to design physics-based animations. Unlike in space-time keyframing or other motion control techniques, our method only requires the user to specify time constraints on the simulation. We explored examples of physics-based motions designed using nonphysical input signals, such as timings from music scores.

2 Related Work

Event-based sound techniques [Takala and Hahn 1992] are widely used in computer graphics to synchronize sound effects, such as "clicks," with animated contact events, and pre-recorded sounds are routinely added by sound designers, or generated by foley artists, to enhance contact events. Unfortunately, complex sounds, especially involving multiple contact events of bouncing, sliding, rolling, chattering, etc., can be difficult to synchronize with pre-generated visual events. Alternately, recent advances in physics-based sound synthesis have enabled the generation of physics-based animations with various synchronized contact sounds [van den Doel et al. 2001; O'Brien et al. 2001; Chadwick et al. 2009; Zheng and James 2010; Chadwick et al. 2012]. Unfortunately, despite perfect synchronization, generating realistic physics-based sound models can be hard for natural sounds, such as a crumpled paper ball, or a thin plastic shell, hitting a specific wooden surface. In contrast, we leverage natural recorded sounds, and explore ways to optimize audiovisual synchronization using motion control.

Inverse-Foley Animation can be viewed as a spacetime optimization problem [Witkin and Kass 1988; Cohen 1992] involving rigid-body motion laws and frictional contact physics, but devoid of other spatial constraints. Unfortunately this motion estimation problem is extremely underconstrained and mathematically ill-posed, which complicates convergence for nonlinear optimization methods. Prior methods for motion design have leveraged the ability of humans to interactively navigate the space-time solution space of rigid-body contact events to obtain desired motions [Popović et al. 2000]. Unfortunately, they do not appear suitable for navigating time-only constraints efficiently, and while their derivative-based search methods could be used to optimize a sequence of contact times, they are inherently local methods whereas future contacts are easily created/destroyed when optimizing initial conditions and contact force perturbations. Motion sketching has been proposed to help animators estimate plausible motions they desire [Popović et al. 2003], which exploits the sketch to provide a good initial guess to help nonlinear optimization methods converge. In contrast, Inverse-Foley Animation is essentially a time-based sketch, which lacks spatial information to help nonlinear optimization.

Random sampling techniques have been used to explore the space of initial conditions and other simulation parameters, that could produce desired outcomes [Tang et al. 1995]. Barzel et al. [1996] introduced the idea of plausibility for animations, arguing that there can be many acceptable simulations. Markov chain Monte Carlo (MCMC) has been used to sample animations satisfying specified constraints [Chenney and Forsyth 2000]. Similar sampling methods can be used to optimize contact-event times, however downsides are that optimization times can be long, and that some methods (such as MCMC) require extensive parameter tuning. In addition, we found that forward sampling methods have a hard time hitting all of the contact event times, necessitating frequent restarts, and that it can be very hard to find plausible contact events that lead to terminal (zero energy) contact states at the desired times using forward search. Sampling reverse-time rigid-body motion might be better suited to this latter case, however the ill-posed nature of reverse-time motion poses its own challenges [Twigg and James 2008], and sampling multiple time-based constraints is still difficult. Many-Worlds Browsing [Twigg and James 2007] allows users to efficiently explore large numbers of sampled rigid-body simulations using a ranking interface, interactive browsing methods, and user-guided adaptive contact-force sampling, but does not provide any specific tools for global optimization of motion synchronization to find those needle-in-a-haystack simulations. In contrast, given an input sound, our contact-event graph can search a contact-event database to estimate synchronized motions. By using blending and time warping, our system can find plausible solutions even when the time constraints are hard (or infeasible) to satisfy.

Our use of contact-event graphs are closely related to "motion graph" techniques used to animate characters using motion-capture databases [Kovar et al. 2002; Lee et al. 2002; Arikan and Forsyth 2002]. Prior work did not explore motion graphs for single rigid bodies since their motion is easy to compute, and other methods exist for rigid-body control. In contrast, we consider a graph of rigid-body contact events to optimize contact times efficiently. Like motion graphs, we also exploit the planar nature of motions in our database to freely orient and position contact events on the plane. There are several differences between contact-event graphs and traditional motion graphs. Motion graphs typically transition between each frame, whereas contact-event graphs transition between contact events, with edges representing rigid-body free flight. Also, motion graph techniques often rely on ensuring the graph is connected, so arbitrarily long streams of motion can be synthesized. In passive rigid-body motion, energy is lost during each contact, so we may not expect to revisit states, and arbitrarily long streams of motion are not possible without accumulating large transition errors.

Synchronization has also been studied in work on aligning animation with sound and music. Cardle et al. [2002] analyzed input MIDI scores, and transformed keyframe animation curves to improve synchronization. Kim et al. [2003] extracted motion beats from input motion data, used these motion beat examples as nodes in a movement transition graph, and traversed the graph to syn-

thesize motion that aligned to an input MIDI signal. Lee and Lee [2005] used a "music graph" and retiming techniques to modify background music and animation curves simultaneously. In constrast, our contact-event graph method is heavily constrained by rigid-body contact physics. We also use time-warping techniques [Bruderlin and Williams 1995] but only to adjust simulated contact-event times to better match an input sound track.

In computer vision, the estimation of ballistic rigid-body motions can be challenging for fast motions and nonsmooth trajectories due to contact [Yilmaz et al. 2006], and physics-based models can improve motion tracking [Duff et al. 2011]. For example, Bhat et al. [2002] estimate ballistic rigid-body motion from video using an optimization method that exploits the smoothness of free-flight motion to track translational and rotational motion. In contrast, we use sound to constrain contact event times of nonsmooth rigid-body motions, possibly with position-level contact constraints from video, but with different yet plausible free-flight motions.

3 Input Contact-Event Specification

Given an input sound or other signal (e.g., MIDI events) for which we wish to find a synchronized animation, we first identify the timing and amplitudes of "contact events" in the audio stream. Unfortunately, identifying contact events robustly and automatically for arbitrary contact sounds, or other creative inputs, is a tricky problem, and we therefore rely on the user to provide the following information to the motion synthesis system.

Contact-time signature of sound: The user provides the following contact-event attributes:

- the target contact event times, $\bar{t}_1, \ldots, \bar{t}_n$, associated with the approximate beginning of each of n significant contact events. Without loss of generality we assume that $\bar{t}_1 = 0$.
- the rough duration of each contact event, $\Delta\bar{t}_k$. "Discrete" contact events, such as bouncing, which have no significant contact duration are given a zero duration. Longer and more complex "continuous" sequences of contact events, such as sliding or rapid sequences of micro-collisions at the end of motions, are given a nonzero duration (see Figure 3).

These times and durations indicate when contact activity should occur in the animation, and are determined by the user listening to the sound, and marking it in a sound visualization tool, such as Audacity. We refer to the list of significant contact event times and their durations as the *contact-time signature*, $\bar{\mathcal{T}} = \{(\bar{t}_k, \Delta\bar{t}_k)\}_{k=1..n}$. In practice, it takes less than a minute to identify contact events for a given input sound. Please see Figure 4 for examples of annotated contact sounds used in our system.

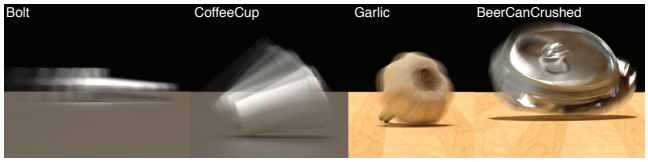


Figure 3: Simulated continuous contact events synchronized to input sounds: *a sliding bolt, a chattering coffee cup, the scruffling of a rolling garlic bulb, and spolling of a crushed beer can.*

The last contact event, n, is flagged as being a *terminal contact state* (i.e., a stable, zero-energy, rest state), or not. If not, then the synthesized motion can "bounce" out of the final contact event, instead of having to come to rest.

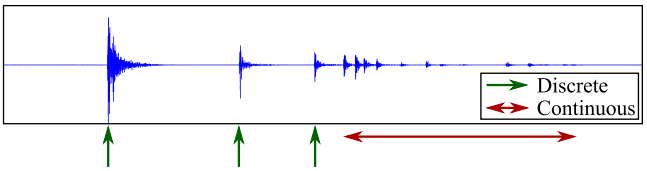


Figure 4: Input sound *with user-annotated contact-event times.*

Contact event amplitudes: Given the user-annotated sound signal, we can automatically estimate contact event amplitudes, $\bar{a}_1$, $\ldots$, $\bar{a}_n$, from the sound signal. For a discrete event at time $\bar{t}_k$, we set the amplitude $\bar{a}_k$ to be that of the nearest peak in a small time window—we use 20 ms in our setup. For continuous events, we use the amplitude values in the continuous region which the user selected. Assuming contact event k is s samples long, we refer to the amplitude in the j^{th} bin as $\bar{a}_k[j]$. To normalize the amplitudes, we divide through by $\bar{a}_1$, such that hereafter we assume that $\bar{a}_1 = 1$ (if the first event is continuous, we normalize by $\max \bar{a}_1[j], j = 1, \ldots, s$).

4 Rigid-Body Contact Problem

We seek to estimate motion for a single dynamic rigid body undergoing a sequence of contact events with a planar environment. In this section we define the rigid-body problem, notation, and its contact-time signature used to optimize synchronization.

Preliminaries: Rigid body dynamics is standard in computer graphics, and we refer the reader to an appropriate reference [Bender et al. 2013]. We will assume that the surface of the object, Γ, is approximated by a triangle mesh. The rigid-body motion is described by the following variables. The body has mass m, and body-space inertia tensor $\boldsymbol{I}$. Let the position of its center of mass be $\boldsymbol{x}$, its orientation given by the quaternion $\boldsymbol{q}$, the linear velocity by $\boldsymbol{v}$, and the angular velocity is denoted by $\boldsymbol{\omega}$. The rigid-body position is $\mathbf{p} = (\boldsymbol{x}, \boldsymbol{q})$, its velocity is $\mathbf{v} = (\boldsymbol{v}, \boldsymbol{\omega})$, and the total state is given by $\psi = (\mathbf{p}, \mathbf{v})$. Gravitational acceleration is $\boldsymbol{g}$. In our implementation, rigid-body motions were simulated using the Open Dynamics Engine (ODE), with contact modeled as a non-penetration constraint and solved as a velocity-level LCP. For planar contact, the convex-hull of each object model was used for rapid simulation.

Contact Events: For simplicity, assume that the environment is a single infinite plane, and that the rigid body undergoes passive motion under gravity that leads to sound-producing contact events with the plane (see Figure 5). We denote the times of these contact events by $t_1, t_2, \ldots, t_n$ for a sequence of n contact events, and again we are free to assume that $t_1 = 0$. These events may be either what we will call "discrete contact" events, such as bouncing impact, or longer "continuous contact" events, such as rolling, sliding, or rapid chattering. In either case, let t_k denote the starting time of the k^{th} event; denote the time immediately before the contact event by t_{k-} and immediately after by t_{k+}. Denote the position at the start of the contact event by $\mathbf{p}_k$, the pre-impact velocity by $\mathbf{v}_{k-} = \mathbf{v}(t_{k-})$, and the post-impact velocity by $\mathbf{v}_{k+} = \mathbf{v}(t_{k+})$. For discrete contact events, such as bouncing, we can often assume without loss of generality that the event is instantaneous, i.e., $t_{k-} \approx t_{k+}$, however, in practice, the contact event may last one or more time steps depending on the type of integrator or contact resolution method used. In contrast, for continuous events, contact forces exist for a longer period of time, with $t_{k-} < t_{k+}$. We denote the rigid-body state immediately before impact by ψ_{k-} and the state immediately after impact by ψ_{k+}.

Contact-time signature of motion: In order to simplify com-

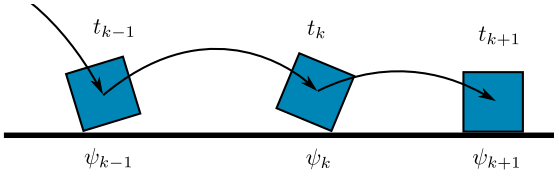


Figure 5: Contact Sequence

parison with other sounds and signals, we use a contact-time signature to summarize the rigid-body animation contact event sequence. Analogous to the contact-time signature for the input sound, $\bar{\mathcal{T}}$, we can define a contact signature for the simulation, $\mathcal{T}$, that consists of the contact start times, t_k, and the contact duration $\Delta t_k = t_{k+} - t_{k-}$. Ideally one could search the simulation space to find motions with matching contact-time signatures, $\mathcal{T} = \bar{\mathcal{T}}$. However, the raw simulation output is noisy, and requires some filtering before direct comparison to the simplified user-specified $\bar{\mathcal{T}}$. Without loss of generality, assume a fixed time step integrator, and consider the sequence of rigid-body contact impulses whose two-norm magnitude is given by $f_i \geq 0$ at time step $i = 1 \ldots N$. Based on this contact impulse data, we can generate an initial noisy version of $\mathcal{T}$, which we then filter as follows (see Figure 6).

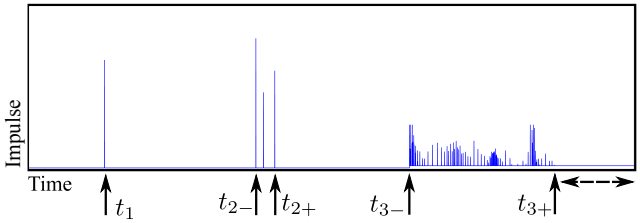


Figure 6: Filtering the simulation's contact-time signature: *There is a discrete event at time t_1. Several close events are grouped into one event between times t_{2-} and t_{2+}. There is a continuous event between times t_{3-} and t_{3+}. The resting contact at the end, denoted by the dashed line, is clipped.*

First, we filter $\mathcal{T}$ to remove small gaps between events by merging adjacent contact-event intervals that are separated by less than a small amount δt; in our implementation, we use δt = 10 ms. For example, if $[t_-, t_+]$ and $[t'_-, t'_+]$ are adjacent (with $t_* < t'_*$), we will merge them to form $[t_-, t'_+]$ iff $t'_- - t_+ < \delta t$. Contact events are renumbered accordingly after a merge. Note that this merging process creates longer "continuous" contact events that contain multiple micro-collision events.

Second, we filter $\mathcal{T}$ to detect discrete (short-lived) impact events. Contact events with duration below a specified duration δt are replaced by zero-duration discrete events: if $\Delta t_k < \delta t$ then we set $\Delta t_k = 0$. In our implementation, we used $\delta t = 10$ ms for most examples; however, we used $\delta t = 30$ ms for the BeerCan, Garlic, and Nut, to ensure the continuous sounds of those objects were treated as continuous events, and not broken up into many microevents.

Third, continuous contact events (with $\Delta t > 0$) can require additional filtering, since final resting states have nonzero contact impulses for which no contact sound will be produced. We therefore detect resting contact states using a velocity criterion, and clip the final resting-contact time interval appropriately, i.e., $[t_{n-}, t_{n+}] \rightarrow [t_{n-}, t'_{n+}]$ for $t'_{n+} < t_{n+}$.

Finally, contact-impulse amplitudes a_k are accumulated along the way for contact events in $\mathcal{T}$. These are computed as the two-norm of all contact impulses f_i associated with each contact event. Given the importance of relative amplitude changes, we normalize all amplitude variations such that $a_1 = \bar{a}_1 = 1$. Note that simulated impulse amplitudes a_k and contact sound amplitudes $\bar{a}_k$ are very different quantities and can not be directly compared, e.g., to require "$a_k = \bar{a}_k$." Given the different nature of impulses and sound, these comparisons can only be used to very roughly specify when large or small impacts occur. Note that this normalization is always computed based on the first event in the motion path, which is not always the first event in the simulation (see § 5).

Synchronized motion problem: The central problem we solve is to find a plausible rigid-body motion which is synchronized with the input sound, in the sense that the sound's $\bar{\mathcal{T}}$ is equal (or close) to the simulation's $\mathcal{T}$, and that the relative amplitude variations are similar. Given that the first contact is always synchronized (since $\bar{t}_1 = t_1 = 0$ and $\bar{a}_1 = a_1 = 1$), we essentially have $n - 1$ remaining contact events to synchronize. The parameters to be optimized are the initial conditions of the initial impact ψ_{1-}, as well as small contact impulses that can perturb each contact event. While other simulation parameters are also uncertain, such as the rigid body's shape, mass, etc., as well as the contact friction and restitution properties, for simplicity we will assume that these are fixed and specified reasonably by the user.

5 Contact-Event Graph Approach

We use a data-driven approach to approximate solutions to the synchronized motion problem. Given the prevalence of flat surfaces in our environments, our method is optimized to find synchronized motions of a single rigid body in a planar environment.

5.1 Sampling Rigid-Body Motions

We construct a database of rigid-body contact events by simulating thousands of rigid-body contact sequences as follows. We randomly sample the pre-contact state of the object, $(\boldsymbol{x}_{1-}, \boldsymbol{q}_{1-}, \boldsymbol{v}_{1-}, \boldsymbol{\omega}_{1-})$, at the time of first contact. Given translation invariance of the planar environment, it is sufficient to select any center-of-mass position $\boldsymbol{x}_{1-}$ above the contact plane. We then sample a random quaternion orientation $\boldsymbol{q}_{1-}$ [Kuffner 2004] (Figure 7a). The object is then pushed down into contact with the plane by finding the minimum-height vertex on the object (Figure 7b). The pre-contact velocities are sampled as follows (Figure 7c). The direction of the linear velocity $\boldsymbol{v}_{1-}$ is importance sampled from the lower hemisphere (so that the velocity is into the plane) with a cosine distribution about the normal used to emphasize more vertical directions[1]; the magnitude $\|\boldsymbol{v}_{1-}\|_2$ is uniformly sampled (we use $[0.5, 4]$ m/s). Angular velocities are nonuniformly sampled from within a 3-ball; we sample a random direction $\hat{\boldsymbol{\omega}_{1-}}$, then uniformly sample the magnitude (we use $[0, 20]$ rad/s) so as to bias the magnitude toward zero to avoid too many motions with rapidly spinning initial conditions. Using these initial pre-contact conditions, the simulation is run forward until the object comes to rest (Figure 7d). To obtain pre-contact ballistic motion to animate motion for $t < t_1$, we integrate the initial state backwards in time until an earlier contact is found (Figure 7e); we simply forward integrate the rigid-body with negated-velocity initial conditions, $(\mathbf{p}_1, -\mathbf{v}_{1-})$. This process is repeated as many times as necessary to create a database of simulations. Note that given translational invariance, this motion sampling problem has 3 rotation and 6 velocity degrees of freedom; further exploiting planar rotational invariance yields an 8-dimensional sampling problem.

[1] For most example objects, dropping was the natural motion, so we only sampled the lower hemisphere up to 30° from vertical. For the Nut and Bolt, where rolling motions were common, we sampled up to 80° from vertical.

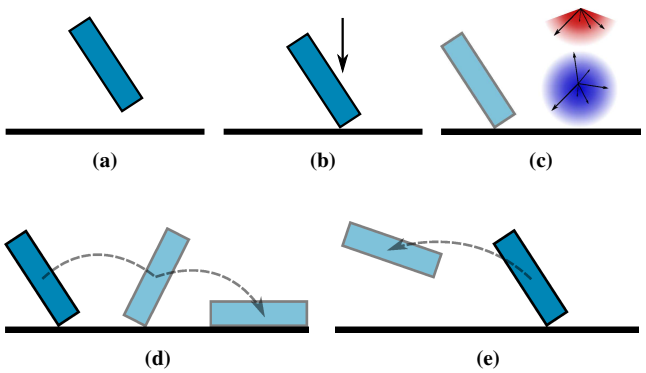


Figure 7: Motion Sampling: *To sample initial simulation parameters, we (a) sample a random orientation, (b) push the object into contact, (c) sample linear (red) and angular (blue) velocities, (d) simulate forwards until the object comes to rest, and (e) simulate backwards to obtain pre-contact ballistic motion.*

5.2 Contact-Event Graph Construction

The motion database is turned into a contact-event graph where each node represents a contact event, and edges represent inter-contact motions that transition between these contact states (see Figure 8). The weight on each edge represents how expensive each transition is, which measures both rigid-body contact state errors, as well as synchronization errors when used for a specific contact-event time.

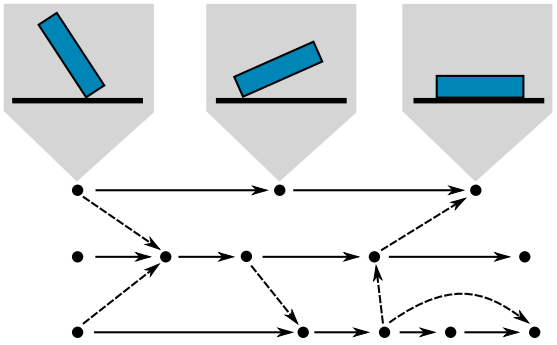


Figure 8: Contact nodes and edges: *Nodes represent contact states, and edges represent transitions between these states. Solid arrows are physically simulated transitions, and dotted arrows are transitions we can make by computing a motion connection.*

For each of the randomly simulated motions, we compute the filtered contact-time signature, $\mathcal{T}$, associated contact states (see §4), and contact amplitudes. Each rigid-body simulation is abstracted as a sequence of contact-event nodes connected by edges which represent connecting trajectories. *Terminal nodes* represent contact states that bring the object to rest, and they are used exclusively to synchronize with terminal events in the input contact-time signature.

5.3 Motion Transitions

Given two motions which have similar contact events (similar post-contact velocity, orientation, etc.), we now describe how to smoothly transition from one motion to the other (see Figure 10).

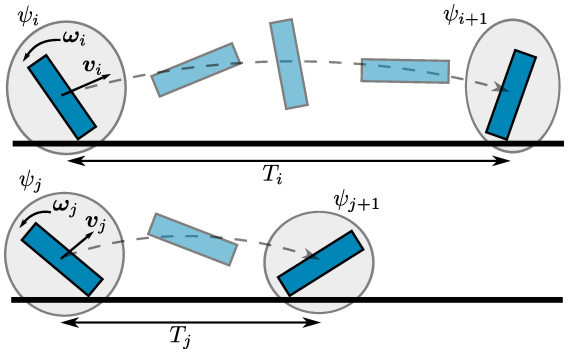


Figure 10: Transitions: *Given two simulated contact sequences $\psi_i \rightarrow \psi_{i+1}$ and $\psi_j \rightarrow \psi_{j+1}$, we can transition from ψ_i to ψ_{j+1} by registering ψ_j to ψ_i and computing a motion connection.*

5.3.1 Registering contact events

To evaluate contact state similarity (for edge weight determination) or to evaluate a motion transition, we exploit translational and rotational invariance on the plane to rigidly register contact events, thereby increasing the fit quality. To register two contact states i and j, we can transform state j to better match i by performing a planar registration of the two contact states at the post-contact times, t_{i+} and t_{j+} (see Figure 9). The rigid transformation involves (1) a 2D translation that best aligns the center of masses of the two bodies, and (2) a planar rotation that best aligns the orientations, $\boldsymbol{q}_{i+}$ and $\boldsymbol{q}_{j+}$. There is an analytical formula for this rotation [Shin et al. 2001].

5.3.2 Motion Connections

Given two similar contact states i and j, which we assume are registered, we can smoothly transition from state i to state $j+1$ by computing a modified rigid-body trajectory that connects the states. However, unless these two motions are *very* similar, directly interpolating the rigid-body trajectories through time, e.g., using rotational slerp or other methods for SE(3) interpolation [Belta and Kumar 2002; Hofer and Pottmann 2004], can introduce nonphysical distortion for motions with very large and/or different angular velocities. Instead, we propose a different approach that involves computing a perturbation to the i^{th} post-contact momentum, so that the resulting motion matches the (registered) position and orientation at $j+1$. In our implementation, we compute momentum perturbations separately for the linear and angular components given the over-constrained nature of the boundary value problem.

Given two registered discrete-event contact states, i and j, we evaluate the ballistic trajectory connecting i to $k = j+1$ by evaluating their linear and angular motions as follows. Without loss of generality, assume that contact events i and j both end at time $t = 0$, and impact k occurs at time T later. In our system we set $T = T_j$.

Linear Motion Connections: Linear blending is relatively simple, and is performed by computing the post-impact linear velocity required for the object's center of mass, starting at $\boldsymbol{x}_{i+}$, to end up at $\boldsymbol{x}_{k-}$ at time T later. It can be computed directly as $\boldsymbol{v}_{blend} = (\boldsymbol{x}_{k-} - \boldsymbol{x}_{i+})/T - T\boldsymbol{g}/2$, and is simply a modified v_{i+}.

Angular Motion Connections: The angular motion connection is slightly more involved. Denote the post-contact angular momentum (which is preserved over the ballistic trajectory) by $\boldsymbol{L}_{i+}$ and $\boldsymbol{L}_{j+}$, respectively. We seek to find an angular motion trajectory that has initial orientation $\boldsymbol{q}_{i+}$ at $t = 0$, and attains the final orientation $\boldsymbol{q}_{k-}$ at $t = T$. There are many possible solutions to this

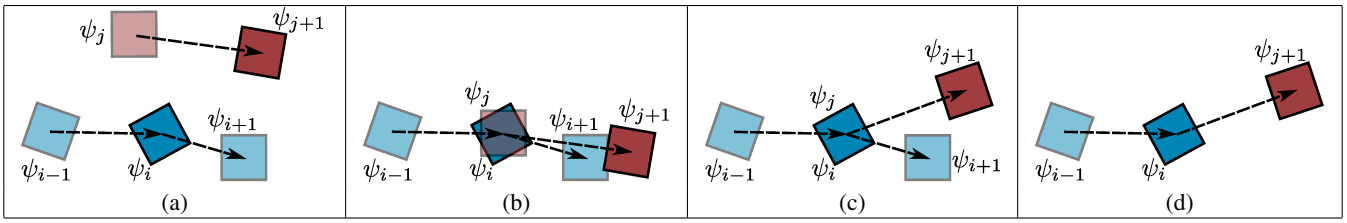


Figure 9: Contact Registration (top down view): *(a) To transition from state ψ_i to state ψ_{j+1}, we register state ψ_j to state ψ_i. (b) First a planar translation is applied to align ψ_j with ψ_i. (c) Then a planar rotation is applied to minimize the orientation error between ψ_i and ψ_j. (d) Then we can evaluate the transition from ψ_i to ψ_{j+1}.*

boundary value problem, and we use a forward shooting method that starts at $\boldsymbol{q}_{i+}$, then seeks to hit $\boldsymbol{q}_{k-}$ while keeping the initial angular momentum close to that of the original motions. Specifically, we seek a free-flight rigid-body angular motion that has an initial angular momentum near the average of the two momenta, $\boldsymbol{L} = (\boldsymbol{L}_{i+} + \boldsymbol{L}_{k-})/2 + \Delta\boldsymbol{L}$, where $\Delta\boldsymbol{L}$ is a correction, such that $||\Delta\boldsymbol{L}||^2_{I^{-1}} = \Delta\boldsymbol{L}^T\boldsymbol{I}^{-1}\Delta\boldsymbol{L} = \Delta\boldsymbol{\omega}^T\boldsymbol{I}\Delta\boldsymbol{\omega}$ is ideally small. We define an endpoint orientation error using the quaternion error function $f(\Delta\boldsymbol{L}) = |\log\left(q(T,\Delta\boldsymbol{L})q_1^{-1}\right)|^2$ where $q(T,\Delta\boldsymbol{L})$ is computed by integrating Euler's angular equations of motion for the rigid body. By linearizing $f(\Delta\boldsymbol{L}) \approx f(0) + \boldsymbol{J}\ \Delta\boldsymbol{L}$, where $\boldsymbol{J} = \frac{\partial f}{\partial \Delta\boldsymbol{L}} \in R^{1\times 3}$ is the Jacobian (which we compute using forward differences), we can write the Newton's method update (with weighting) as $\Delta\boldsymbol{L} = W\boldsymbol{J}^T\boldsymbol{\lambda}$, then determine $\boldsymbol{\lambda}$ so that $f = 0$, and thus obtain $\Delta\boldsymbol{L} = -W\boldsymbol{J}^T(\boldsymbol{J}W\boldsymbol{J}^T)^{-1}f(0)$. The weighting matrix W is set to the inertia matrix $\boldsymbol{I}$, thus giving extra momentum to heavy axes. When doing the Newton step, we damp the updates to avoid large angular changes which could lead to instabilities. We use a simple model to reason about angular changes, $\Delta\theta \approx ||\Delta\boldsymbol{\omega}||T = ||\boldsymbol{I}^{-1}\Delta\boldsymbol{L}||T$; in our implementation, the $\Delta\boldsymbol{L}$ update is scaled so that $\Delta\theta$ is less than 10^o. In rare cases where Newton's method does not converge, we simply discard the edge.

Time Warping: Any transition edge can be slightly re-timed to match the target inter-contact time T using time warping [Bruderlin and Williams 1995], however in practice only very limited retiming is used since this can lead to significant distortions. In practice we are able to achieve target times by keeping retiming effects less than 30%. Note that any retiming is done *after* blending to avoid retiming the equations of motion.

5.4 Edge weights for inter-contact transitions

Transition errors can be divided into two types: (1) static errors for motion, which include velocity and orientation differences; and (2) dynamic errors for sound synchronization, which include retiming, amplitude errors, and whether the event is terminal or not. Static errors depend only on the contact events from the database, whereas dynamic errors are input-sound related and depend on the target contact signature being matched.

Transition quality is measured by a factored edge weight model:

$$w = f_v f_q f_t f_s f_E f_{end} \tag{1}$$

where the affinity factors f_* correspond to velocity match (f_v), quaternion/orientation match (f_q), time warping to ensure synchronization (f_t), contact sound/force similarity (f_s), energy limiter (f_E), and terminal events (f_{end}). Higher edge weights correspond to better transitions. We now describe these factors, and the tuned parameters used in all of our examples.

Post-contact velocity similarity is assessed based on the simulated velocity change $\Delta\boldsymbol{v}_i = \boldsymbol{v}_{i+} - \boldsymbol{v}_{i-}$ (see Figure 11), and $\Delta\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i+} - \boldsymbol{\omega}_{i-}$. We define plausibility parameters which represent

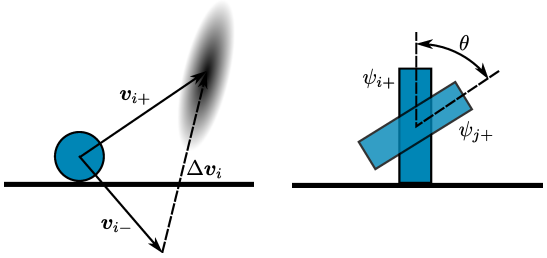


Figure 11: Static state comparison: *(Left) The plausible region for velocities depends on the magnitude and direction of $\Delta\boldsymbol{v}_i$. (Right) Orientations are compared based on the smallest angle of rotation between the two states.*

the maximum plausible angle change for the linear (θ_L) and angular (θ_A) velocities, and the maximum plausible relative magnitude change for the linear (ϕ_L) and angular (ϕ_A) velocities. Inspired by O'Sullivan et al. [2003], we use $\theta_L = 20^o$, $\phi_L = 0.4$, and $\phi_A = 0.2$; for θ_A, we set $\theta_A = 10^o$. We evaluate the velocity similarity between two states using a weighted squared magnitude. We first compute rotations R_L and R_A, which rotate $\Delta\boldsymbol{v}_i$ and $\Delta\boldsymbol{\omega}_i$ to $(1,0,0)$, respectively, and compute

$$\mathbf{v}_{eval} = \begin{pmatrix} R_L(\boldsymbol{v}_{blend} - \boldsymbol{v}_{j+}) \\ R_A(\boldsymbol{\omega}_{i+} - \boldsymbol{\omega}_{j+}) \end{pmatrix}.$$

We evaluate the magnitude of $\mathbf{v}_{eval}$ using a scaling matrix which sets the magnitude to 1 at the edge of the aforementioned plausible region: defining the diagonal matrix, $\boldsymbol{C}^{-1} = \text{diag}(a,b,b,c,d,d)$,

$$\begin{aligned} a &= \phi_L\|\Delta\boldsymbol{v}_i\| & b &= \phi_L\|\Delta\boldsymbol{v}_i\|\ \tan(\theta_L) \\ c &= \phi_A\|\Delta\boldsymbol{\omega}_i\| & d &= \phi_A\|\Delta\boldsymbol{\omega}_i\|\tan(\theta_A) \end{aligned} \tag{2}$$

we set

$$f_v = e^{-\|\boldsymbol{C}\mathbf{v}_{eval}\|^2}. \tag{3}$$

The Gaussian halfwidth occurs at $\|\boldsymbol{C}\mathbf{v}_{eval}\|^2 \approx 0.69$, and so values outside our "plausibility" region ($\|\boldsymbol{C}\mathbf{v}_{eval}\|^2 > 1$) are unlikely.

Orientation error is important because it affects how much blending will be necessary to perform the transition. The exact plausibility parameter is difficult to quantify, due to the non-linearity of the equations of motion, as well as the fact that the amount of blending required also depends on the velocity similarity and length of the transition. The amount of acceptable orientation difference also depends on the length of the transition, since large differences for short transitions could require large connection impulses. We examine the similarity as a rate of angle difference per unit time. Letting $\theta = |\log(\boldsymbol{q}_i\boldsymbol{q}_j^{-1})|$, we set

$$f_q = e^{-c_q \max\left(\frac{\theta^2}{T^2}, \frac{\theta^2}{T_0^2}\right)} \tag{4}$$

where $c_q = 0.85$ and $T_0 = 0.2$ (this sets the Gaussian halfwidth to $\frac{\theta}{T} = 0.9$ when $T < T_0$, and limits the halfwidth to $\frac{\theta}{T_0} = 0.9$ when $T \geq T_0$, which is about 10^o).

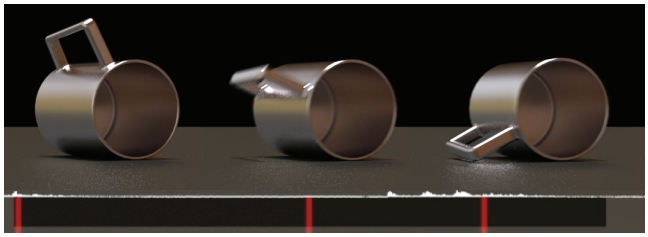


Figure 12: A continuous contact event (MetalCup desk 02)*: Using the sound amplitude (shown at bottom) and force amplitude centroids, we not only synchronize its initial impact (Left), but, as the mug rolls (Middle), also the later handle impact (Right).*

Time warping improves synchronization, and its excessive use is penalized by the f_t factor. Given the goal duration of the inter-contact trajectory, T^*, and the blended trajectory time, T_b, we use

$$f_t = e^{-c_t T_{eval}^2} \quad \text{where} \quad T_{eval} = \left(\frac{\max(T^*, T_b)}{\min(T^*, T_b)} - 1 \right), \quad (5)$$

with $c_t = 100$. This sets the Gaussian halfwidth to 7% retiming, and ensures that retiming errors beyond 15% are unlikely.

Target event sounds and contact forces are compared using 3 weights. Significant differences in relative amplitudes of the contact sound $\bar{a}$ and the simulated contact force a are penalized using the amplitude factor

$$f_a = e^{-c_a |\bar{a} - a|^2} \quad (6)$$

where $c_a = 2.78$, which sets the halfwidth to 0.5. The difference in continuous event lengths, Δt_1 and Δt_2, is measured using

$$f_l = e^{-c_\Delta |\Delta t_1 - \Delta t_2|^2}; \quad (7)$$

we use $c_\Delta = 7000$, setting the halfwidth to 0.01 seconds. To better synchronize amplitudes within continuous events, e.g., near the start or the end (see Figure 12), we match the centroid time of the sound event amplitude distribution, $a[k]$, (or contact force amplitude, $\bar{a}[k]$) as $\tau_a = (\sum_{k=1}^{n} a[k] k \Delta t)/(\sum_{j=1}^{n} a[j])$ (similarly, $\tau_{\bar{a}}$ for $\bar{a}[k]$). Given the event's sound and force centroid times, τ_a and $\tau_{\bar{a}}$, respectively, we compare them using the factor

$$f_c = e^{-c_\Delta |\tau_a - \tau_{\bar{a}}|^2}. \quad (8)$$

The sound comparison affinity factor is the product of the 3 weights,

$$f_s = f_a f_l f_c. \quad (9)$$

For discrete contact events (where $\Delta t \approx 0$), only f_a contributes.

Spurious energy gains are avoided by further restricting transitions. The energy of a state immediately before (-) or after (+) the i^{th} contact event is given by

$$E(\psi_i^{\pm}) = \frac{1}{2} \left(m \|\boldsymbol{v}_i^{\pm}\|^2 + (\boldsymbol{\omega}_i^{\pm})^T \boldsymbol{I} \boldsymbol{\omega}_i^{\pm} \right) + mgh_i^{\pm}$$

where $h_i^{\pm}$ is the height of the object's center of mass. To discourage transitions to higher energy states[2] we set

$$f_E = \begin{cases} 1, & E(\psi_i^-) > E(\psi_j^+), \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

[2]We found that requiring $E(\psi_i^+) > E(\psi_j^+)$ can be too restrictive, especially if the rigid-body model's restitution value is too low.

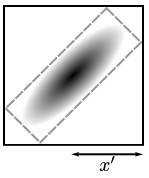
Terminal nodes are used for the final contact event if the object is required to come to rest, and this is achieved by the f_{end} factor: if we are currently considering transitions from event m in the target sequence $\bar{\mathcal{T}}$, then

$$f_{end} = \begin{cases} 1, & \text{if } (m \text{ not terminal}) \text{ XOR } (\text{state } j+1 \text{ is terminal}), \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

5.5 Searching for Synchronized Motions

Walks in the graph correspond to synthesized rigid-body contact sequences. Edge weights are defined such that larger edge weights correspond to good transitions. *We define the path quality (or score) as the product of the edge weights*, instead of the sum. Consequently a motion path can obtain a bad (near zero) score if there exists one rather implausible transition (with near zero edge weight). Given the large search space, we find walks in the graph using a branch and bound technique similar to [Kovar et al. 2002], utilizing a k-d tree to quickly find the closest (best transition) events to the event we are currently at. If the input time signature has only one event, we can simply select a terminal event and its incoming motion. More generally, for $n > 1$ contact events, we find a feasible node synchronized with the second contact event at $\bar{t}_2$ by considering all events i, where T_{i-1} is close to $(\bar{t}_2 - \bar{t}_1)$. Subsequent contact events are found by considering sub-paths of bounded search depth, e.g., the next 5 contact events, using a branch and bound technique. Edges are explored in order of weight, with the best ones being explored first. Given the best bounded-depth sub-path, we retain only the next two contact events, advance the contact-event search horizon, and repeat the process. As we approach the n^{th} terminal node, we restrict the search to include feasible subpaths, i.e., ones that have "terminal nodes" at the n^{th} contact event.

Finding nearest neighbors: We use two 12-dimensional k-d trees to quickly find potential transitions. One of the trees stores terminal events, the other stores non-terminal events. For an event j, the dimensions are $T_{j-1}, \Delta t_j, \mathbf{v}_{j-1+}$ (6 dimensions), the three angles $(\theta_1, \theta_2, \theta_3)$ that the rigid body's principal axes make with the contact normal, and the centroid time τ_j of the contact event. Denoting a point in the tree by $p = (p_1, \ldots, p_{12})$, we use a scaled L_2 distance when searching for nearby points: $dist(p, p') = \sum_{i=1}^{12} s_i^2 (p_i - p_i')^2$, with the scaling factors s_i set to $\frac{1}{halfwidth}$ of the corresponding edge weight affinity factor. When searching for neighbors of event j, the coordinate distance for the inter-contact time (T) is normalized by T_{j-1}, and the principal axes distances are normalized by $\min(T_{j-1}, 0.2sec)$. The scaling of the velocities is slightly more involved because the plausible region is based on Δv_{j-1}, which is not always axis aligned. We first create a bounding box of the quadric described in (3). The box is centered at $\mathbf{0}$, and has edge half-lengths a, b, b (see (2)). The bounding box is rotated by R_L^{-1}, which aligns it with $\Delta \boldsymbol{v}_{j-1}$. The axis-aligned bounding box of the rotated original bounding box is computed, and has edge half-lengths x', y', z'. These half-lengths are then used to normalize the velocity distances. Angular velocities are treated similarly. Note that the k-d trees only need to be constructed once. The scaling in the distance function allows the distances to be adjusted depending on the query point.



Motion Score: The optimizer searches for paths that maximize the product of edge weights. To report a normalized motion score for humans (independent of the number of contacts), we define a motion's Score as the geometrically averaged affinity factor values

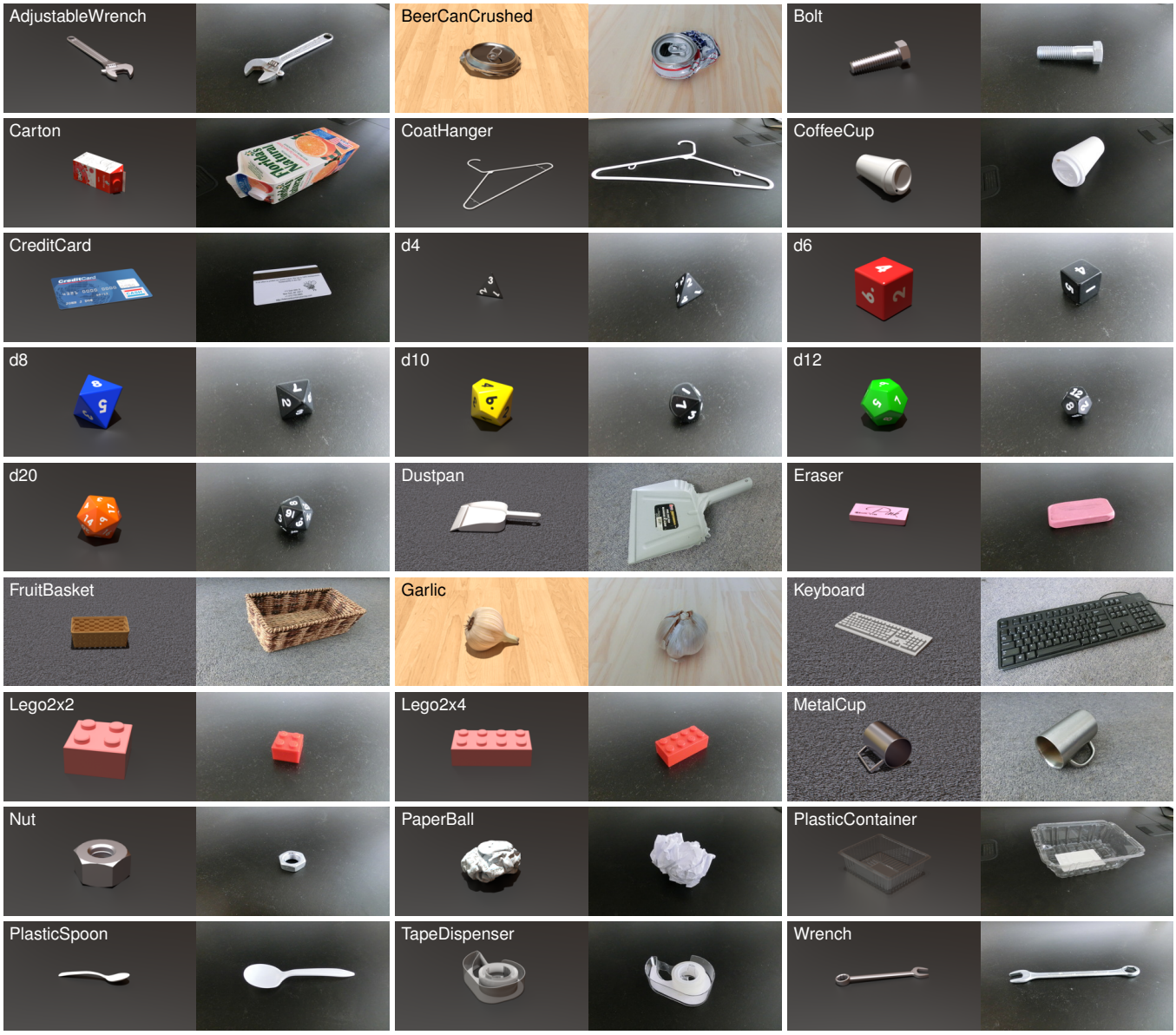


Figure 13: Virtual models (left) and real-world objects (right) *used in over 434 IFA experiments.*

over the n-contact motion sequence,

$$\text{Score} = \left(\prod_i f_v f_q f_t f_a f_l f_c\right)^{\frac{1}{6n}}. \tag{12}$$

To shed light on motion quality, we also report the Score independent of the sound amplitude factors,

$$\text{Score}_{\text{w/o sound}} = \left(\prod_i f_v f_q f_t\right)^{\frac{1}{3n}}. \tag{13}$$

Optional speedups: Since searching large contact-event graphs can be slow, we use several optional speedups. To avoid spending time exploring obviously poor transitions, we only explore edges where $\log(f_v) > -10$ and $\log(f_q) > -10$. Furthermore, we use the k-d tree to quickly find and search only the best 5 children/transitions of each node, thereby reducing the number of transitions that must be considered during the branch-and-bound search. For each recorded sound, we also use an "early exit" condition, that terminates the search (and returns the found motion) if the Score is sufficiently high; in our examples, we "early exit" if Score ≥ 0.3. We also enforced a maximum search time of 1 hour to time out on potentially infeasible problems.

Lazy evaluation of blends: We do not actually compute blends for edges during the graph search, to avoid the cost of computing blends for non-simulation edges in the graph, which is potentially very high, e.g., in graphs with hundreds of thousands of edges can require many hours of Newton solves. In practice, we only require blends for edges used in the final animation. Some transitions can introduce ground interpenetration when blending sufficiently dissimilar motions. Since interpenetration can be perceptually bothersome it is not allowed. We initially assume that all edges are feasible, and then once we find an optimal sub-path we compute its blends, then if any edges are infeasible we discard them and recompute the optimal sub-path. In practice, blending costs are reduced enormously, and the search is still fast since very few edges are infeasible. We note that blends depend only on the motions, and not the input sound. While they could be precomputed, this

would require much longer precomputation and more storage for the database.

6 Results

We recorded multiple sounds from 27 real-world rigid objects on multiple surfaces, for a total of 434 sound recordings. For each object, a virtual proxy model was created, with the geometry, mass properties, and contact restitution behavior set to produce a similar likeness. The objects and models are shown in Figure 13. Our system was able to synthesize plausible synchronized motions for all of the objects, and almost all of the sounds. Please see the accompanying video for all audiovisual results and Scores. Statistics are provided in Table 1; all timing statistics are run on a machine with four Intel Xeon 7560 processors at 2.27GHz. Note that *we use the same hand-tuned edge-weight parameters in all cases,* so no extra parameter tuning was required.

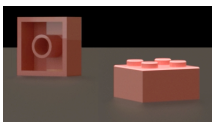
Rigid-body Model	nodes	n	success	search	Score
AdjustableWrench (desk)	2556981	5.5	13 / 13	0.9	0.43
BeerCanCrushed (wood)	1462089	2.9	15 / 15	0.6	0.35
Bolt (desk)	2604880	5.6	7 / 7	14.5	0.25
Carton (desk)	1282526	4.1	25 / 29	1.3	0.32
CoatHanger (desk)	4759056	6.4	18 / 19	9.3	0.46
CoffeeCup (desk)	5710615	7.1	15 / 16	38.2	0.24
CreditCard (desk)	1965526	4.4	22 / 23	6.8	0.44
d4 (desk)	3750746	15.7	6 / 10	62.7	0.17
d6 (desk)	6751058	20.4	13 / 14	12.1	0.41
d8 (desk)	4867818	22.0	6 / 12	53.9	0.27
d10 (desk)	7623610	25.7	11 / 14	18.6	0.38
d12 (desk)	9934621	26.0	4 / 7	24.2	0.39
d20 (desk)	9427739	26.8	9 / 10	37.7	0.31
Dustpan (wood)	4520601	6.7	6 / 6	21.9	0.37
Dustpan (carpet)	2629017	2.5	4 / 6	6.5	0.21
Eraser (desk)	2003606	4.6	47 / 48	2.1	0.50
FruitBasket (wood)	1697917	4.6	7 / 7	0.3	0.45
FruitBasket (carpet)	1705726	4.0	7 / 10	0.4	0.43
Garlic (wood)	876191	4.5	6 / 6	2.6	0.24
Keyboard (carpet)	2366495	5.9	8 / 9	11.4	0.30
Lego2x2 (desk)	4065133	9.3	9 / 9	16.3	0.36
Lego2x4 (desk)	3994256	8.0	23 / 23	8.8	0.41
MetalCup (desk)	4252157	12.8	6 / 6	22.1	0.32
MetalCup (carpet)	2378823	3.4	9 / 10	3.3	0.26
Nut (desk)	3790909	8.3	9 / 10	12.1	0.33
PaperBall (desk)	3160375	4.6	16 / 20	5.9	0.24
PlasticContainer (desk)	1797494	4.8	18 / 22	2.4	0.32
PlasticSpoon (desk)	1115891	4.6	10 / 15	2.6	0.38
TapeDispenser (wood)	2726872	6.4	10 / 10	1.8	0.41
TapeDispenser (desk)	3561017	7.9	17 / 17	10.8	0.36
Wrench (desk)	4862720	15.7	9 / 11	47.8	0.17

Table 1: Statistics: *For each rigid-body model, a database of 400,000 rigid-body simulations were used to construct a graph with the indicated number of contact-event* **nodes***. Multiple input sounds are used, with the average number of contact events (n). Motions were successfully estimated for most sounds, with the average search time (in* min*) for these successful cases reported (***search***) along with the average* **Score***. Harder examples often having many contacts (such as the dice), longer search times, and a lower score. "Early exits" terminated searches that achieved* **Score** >0.3.

High-speed video comparisons: For some input contact sounds, we also recorded high-speed video of the associated contact experiments (AdjustableWrench, CreditCard, Eraser, d6, Lego2x4, and PaperBall). Although the real and synthesized motions are different, these videos clearly demonstrate the level of synchronization attained. For some examples contact deformations are also visually apparent, e.g., the CreditCard.

Orientation constraints: Adding constraints to the search is helpful, especially for objects that have distinct sounds depending on the type of contact event. This is easily accomplished in our system by adding additional edge weights.

We demonstrate this by adding orientation constraints to the Lego, which has distinct sounds depending on whether its open side lands face down or not, and the Keyboard, which we always wanted to land with the keys facing up. Using the normal vector of a face, $\mathbf{n}$, and a goal direction, $\mathbf{d}$, we construct an affinity factor based on the angle between the two directions, $f_o = e^{-c_o\theta^2}$, where $\theta = \cos^{-1}\left(\frac{\mathbf{n}\cdot\mathbf{d}}{\|\mathbf{n}\|\|\mathbf{d}\|}\right)$, and $c_o = 140$ (which sets the halfwidth to $\theta = 0.0698$ rad $\approx 4^o$). For each contact node, if there is an orientation constraint for that node, we add the corresponding factor to the incoming edge weight. We matched all specified final orientations.



Position constraints and audio-visual capture: Another possible use case of IFA is approximate video-based motion capture. We captured several rigid-body motion sequences with a calibrated overhead camera, and labeled the approximate center of mass of the object at the start of each contact event, giving a desired position (in the plane) $\mathbf{b}_i$ at each contact event i. For simulated contact events, let $\boldsymbol{x}^p$ denote the position projected onto the plane. Motions with only one event could just be translated along the plane to match any position constraint. For longer motions, we align $\boldsymbol{x}^p_{1-}$ with $\mathbf{b}_1$, and rotate the motion to align the first edge $(\boldsymbol{x}^p_{2-} - \boldsymbol{x}^p_{1-})$ with $(\mathbf{b}_2 - \mathbf{b}_1)$. For each subsequent edge, we add an affinity factor based on the position of each contact event, $f_p = e^{-c_p||\boldsymbol{x}^p_{i-} - \mathbf{b}^p_i||^2}$, with $c_p = 7000$ (which sets the halfwidth to 1cm). Audiovisual results are in the accompanying video and Figure 14, and error statistics are in Table 2.

Motion Example	max (cm / relative)	average (cm / relative)	L (cm)
Lego2x4 1	6.0 / .28	3.7 / .18	21.3
Lego2x4 2	7.4 / .18	4.1 / .1	41.1
Lego2x4 3	2.8 / .12	1.9 / .08	24.0
Garlic 1	8.3 / .09	4.4 / .05	96.8
Garlic 2	4.1 / .04	2.4 / .03	96.7
TapeDispenser 1	8.6 / .12	4.5 / .06	71.5
TapeDispenser 2	10 / .15	5.8 / .09	64.8
TapeDispenser 3	7.1 / .18	4.9 / .12	39.6

Table 2: Position Constraint Statistics: *For each motion, we report the maximum and average position errors. Errors are given in cm, as well as units relative to the longest edge L of the motion's bounding box.*

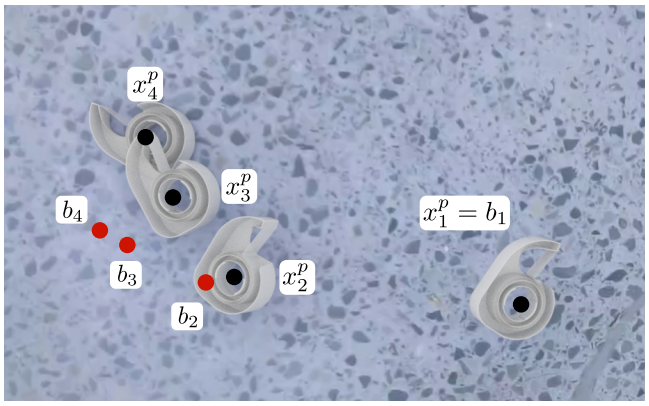


Figure 14: Audio-visual motion estimation of rigid-body contact: *We estimate virtual rigid-body motions with planar object positions $\{\boldsymbol{x}^p_i\}$ (black dots) at contact events that approximate video-recorded object motions with indicated contact positions $\{\mathbf{b}_i\}$ (red dots).*

Music inputs: We also experimented with motion synthesis for highly nonphysical inputs taken from musical scores. Obtaining

plausible synchronized motions can be challenging for such inputs due to infeasibility. For example, it is highly unlikely that a falling object would bounce indefinitely, or tap out a James Brown song. By relaxing plausibility and energy constraints, we were able to synchronize to music. We include two whimsical examples: (1) a 6-sided die tapping out the classic knocking rhythm "*Shave and a haircut. Two bits.*" (see video); and (2) we individually synthesized an ensemble of dice that tap to a popular drum track (see Figure 15). Terminal events were not imposed on any input contact-time signatures. To match longer input sequences, or ones with impossibly long pauses, we allow energy gains by modifing the energy growth factor f_E to allow a user-specified bounded energy gain.

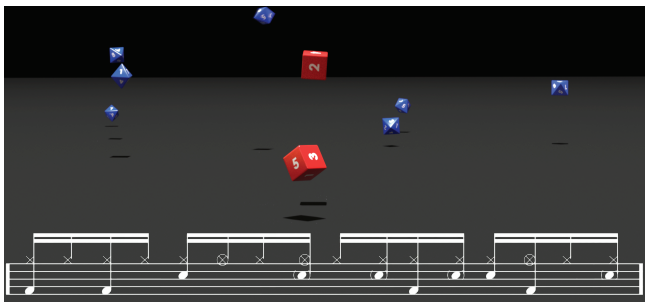


Figure 15: Funky Dice: *Each die taps out drum tracks from the "Funky Drummer" rhythm; blue dice hit "high hat" sixteenth notes, and red dice hit snare and bass notes.*

Database size dependence: We performed several experiments to evaluate the effect of the motion database size on the quality of the synthesized motion. For four of our objects, the CoatHanger, d6, Eraser, and Lego2x2, we synchronized motions using different sized databases. For each database size, we ran the search for 1 hour, then returned the best motion. Results are shown in Figure 16. The motion quality tends to increase slowly with database size, although it is not monotonic due to the approximate nature of the search. Also note that for the d6, the search is unable to find a solution for the small database sizes.

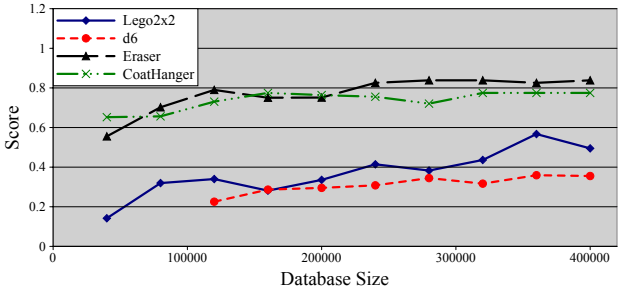


Figure 16: Score vs Database Size: *We searched databases of varying sizes (# simulations) and compared the score (average factor weight) of the resulting motion (the same sound was used for all sizes). For each size, we ran the search for 1 hour (without any early stopping condition). For all four objects, the quality of the solution generally increases with database size. The* d6 *fails to find a solution when using small databases. The* CoatHanger, Eraser, *and* Lego2x2 *found a solution with all the sizes we tested.*

7 Conclusion

We have introduced Inverse-Foley Animation, a new technique for synthesizing rigid-body motions that are synchronized with pre-recorded sounds, or other temporal input signals. By optimizing motion for synchronization, we are able to capture the diversity and richness of real sounds, while avoiding the complexities and limitations of sound synthesis for such sounds. Inverse-Foley Animation has been successfully used to synthesize synchronized motions for dozens of objects, and hundreds of contact sound sequences. Furthermore, such digital techniques, if successful, have the potential to allow sound to be used earlier in the creative animation pipeline, potentially directing and improving the animated events, as opposed to having sound added "after the fact" in post production, and not achieving complete synchronization. For nonphysical input sounds, such as musical scores, IFA is a new tool for creative motion content generation.

Limitations and Future Work: Planning synchronized physics-based motions is particularly challenging. Perhaps the biggest limitation of our method is that, even for real-world contact sound inputs, it is not guaranteed to find a plausible solution. For a small fraction of the real-world contact-sound inputs, our system was not able to find a motion with a satifactory Score, which can be due to several factors: rigid-body dynamics modeling error (differing geometry, mass properties, friction, or (most commonly) contact restitution); violation of the rigid-body assumption, e.g., the CreditCard deforms noticeably on contact, can lead to different contact behavior; under-sampled motion databases can introduce infeasibility; even for large databases, the graph search method is not guaranteed to find solutions due to the incomplete nature of its branch-and-bound search—an issue for very long contact sequences, such as the dice (d4, d6, ..., d20). Other limitations arise from the fact that the input contact-time signature is a gross simplification of temporal contact dynamics, and hand labeling of contact times solely from sound can be ambiguous. For some of the sounds, the annotated contact signature may be sufficiently incorrect, and impossible for the model to satisfy plausibly, even without rigid-body modeling limitations. Some contacts may be too quiet to hear, and fast contacts (e.g., when dice are chattering) may be too close to each other for humans to distinguish. Our system is not highly optimized for speed, and significant improvements could be made, e.g., to accelerate graph-based motion synthesis. Our approach has many motion design parameters, which we have hand tuned, and although we use the same parameters for our examples, better results may be obtained by optimizing them further. Finally, this work investigated the motion of a single rigid-body on a plane, and it is interesting to consider the more general problem of how to hallucinate animations that match sound. Future work should consider non-planar environments and multiple interacting objects, as well as other motion phenomena, such as characters, non-rigid bodies, and fluids, and other nonphysical temporal inputs. Finally, physics-based methods that use both audio and visual inputs streams are needed for tracking the motion of contacting rigid bodies.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. Mitsuba was used for all renderings. We acknowledge funding and support from the National Science Foundation (HCC-0905506, DGE-1144153), Intel, the John Simon Guggenheim Memorial Foundation, and donations from Pixar and Autodesk. This research was conducted in conjunction with the Intel Science and Technology Center–Visual Computing. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or others.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion

generation from examples. *ACM Transactions on Graphics 21*, 3 (July), 483–490.

BARZEL, R., HUGHES, J. F., AND WOOD, D. N. 1996. Plausible motion simulation for computer graphics animation. In *EGCAS 96: Seventh International Workshop on Computer Animation and Simulation*, 183–197.

BELTA, C., AND KUMAR, V. 2002. An SVD-based projection method for interpolation on SE(3). *Robotics and Automation, IEEE Transactions on 18*, 3, 334–345.

BENDER, J., ERLEBEN, K., AND TRINKLE, J. 2013. Interactive simulation of rigid body dynamics in computer graphics. In *Computer Graphics Forum*, Wiley Online Library.

BHAT, K. S., SEITZ, S. M., POPOVIĆ, J., AND KHOSLA, P. K. 2002. Computing the physical parameters of rigid-body motion from video. In *Computer VisionECCV 2002*. Springer, 551–565.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 97–104.

CARDLE, M., BARTHE, L., BROOKS, S., AND ROBINSON, P. 2002. Music-driven motion editing: local motion transformations guided by music analysis. *Proceedings 20th Eurographics UK Conference*, 38–44.

CHADWICK, J. N., AN, S. S., AND JAMES, D. L. 2009. Harmonic Shells: A Practical Nonlinear Sound Model for Near-Rigid Thin Shells. *ACM Transactions on Graphics 28*, 5 (Dec.), 119:1–119:10.

CHADWICK, J. N., ZHENG, C., AND JAMES, D. L. 2012. Precomputed acceleration noise for improved rigid-body sound. *ACM Transactions on Graphics 31*, 4 (July), 103:1–103:9.

CHENNEY, S., AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. *Proceedings of the 27th annual conference on*, 219–228.

CHION, M. 1994. *Audio-Vision: Sound on Screen*. Columbia University Press.

COHEN, M. 1992. Interactive spacetime control for animation. In *ACM SIGGRAPH Computer Graphics*, ACM, vol. 26, 293–302.

DUFF, D. J., MORWALD, T., STOLKIN, R., AND WYATT, J. 2011. Physical simulation for monocular 3d model based tracking. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 5218–5225.

HOFER, M., AND POTTMANN, H. 2004. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics 23*, 3 (Aug.), 284–293.

KIM, T.-H., PARK, S. I., AND SHIN, S. Y. 2003. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics 22*, 3, 392–401.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion Graphs. *ACM Transactions on Graphics 21*, 3, 473–482.

KUFFNER, J. 2004. Effective sampling and distance metrics for 3D rigid body path planning. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, IEEE, 3993–3998.

LEE, H.-C., AND LEE, I.-K. 2005. Automatic synchronization of background music and motion in computer animation. *Computer Graphics Forum 24*, 3, 353–361.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics 21*, 3 (July), 491–500.

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

O'SULLIVAN, C., DINGLIANA, J., GIANG, T., AND KAISER, M. K. 2003. Evaluating the visual fidelity of physically based animations. *ACM Transactions on Graphics 22*, 3 (July) (July), 527–536.

POPOVIĆ, J., SEITZ, S., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press/Addison-Wesley Publishing Co., 209–218.

POPOVIĆ, J., SEITZ, S. M., AND ERDMANN, M. 2003. Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics 22*, 4 (Oct.), 1034–1054.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Trans. Graph. 20*, 2 (Apr.), 67–94.

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

TANG, D., NGO, J. T., AND MARKS, J. 1995. N-body spacetime constraints. *The Journal of Visualization and Computer Animation 6*, 3 (July-Sept.), 143–154.

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.

VAN DEN DOEL, K., KRY, P. G., AND PAI, D. K. 2001. Foleyautomatic: Physically-based sound effects for interactive simulation and animation. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 537–544.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Computer Graphics (Proceedings of ACM SIGGRAPH '88)*, ACM, vol. 22, 159–168.

YILMAZ, A., JAVED, O., AND SHAH, M. 2006. Object tracking: A survey. *ACM Computing Surveys 38*, 4, 13.

ZHENG, C., AND JAMES, D. L. 2010. Rigid-Body Fracture Sound with Precomputed Soundbanks. *ACM Transactions on Graphics 29*, 4 (July), 69:1–69:13.