Procedurally-Generated Audio for Soft-Body Interactions

by

Feng Su

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

Procedural audio generation is an important method for automatically synthesizing realistic sounds for computer animations and games. While synthesis techniques for rigid bodies have been well studied, few existing works have tackled the challenges of soft-body interactions. In this dissertation, we explore practical methods for procedural audio generations of soft bodies. First, we synthesize both impulsive and continuous sounds for elastic deformations. Our method is built on granular synthesis that retargets sound tracks of real-world recordings to match the motion of input elastic objects. Next, to synthesize sounds for plastic deformations, we introduce a concatenative synthesis method with a fast feature correlation technique, which is able to calculate the motion events of highly deformable objects at run time and simultaneously generate sounds according to these motion signals. Last but not least, we focus on evaluating the synthesized audio using both subjective and objective techniques, and the results demonstrate that our proposed synthesis methods can produce convincing sounds for soft bodies that are comparable to the recorded ones.

Our presented methods do not require computationally expensive physics simulations and have improved previous data-driven synthesis approaches with more efficient analysis, control and evaluation techniques, which makes it possible to automatically generate plausible audio for a variety of soft-body interactions.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Chris Joslin, for his vision and guidance on this research as well as my career goal, and for his generosity and encouragement throughout both my Master's and PhD studies. He has always been a superb mentor for me.

I am also grateful to my thesis committee members for their constructive suggestions and feedbacks that helped me improve my work tremendously.

Many thanks to my friends and office mates during my graduate life, who offered me helps and enriched my experience along the way. I will always remember the laughs, funs and gatherings we had together.

Finally, my sincere appreciation goes to my family, my parents and grandparents, for their love and supports no matter what, even though I'm far from home for many years; and to my wonderful girlfriend Xinhe, who have had an incredible impact on my life: Thank you so much for your endless love, support and understanding!

Table of Contents

List of Tables

List of Figures

List of Acronyms

ANOVA	Analysis of Variance
CC0	Creative Commons
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSS	Concatenative Sound Synthesis
DDR3	Double Data Rate 3
FDTD	Finite-Difference Time-Domain
FEM	Finite Element Method
FFAT	Far-Field Acoustic Transfer
FID	Fréchet Inception Distance
FPS	Frames Per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HRTF	Head-Related Transfer Function
IS	Inception Score
KL	Kullback-Leibler
LMA	Linear Modal Analysis
MFCC	Mel-Frequency Cepstral Coefficient
MIDI	Musical Instrument Digital Interface
MOS	Mean Opinion Score

MPS	Mean Preference Score
ODE	Ordinary Differential Equation
PAN	Pre-computed Acceleration Noise
PAT	Pre-computed Acoustic Transfer
PCG	Procedural Content Generation
PhM	Physical Modeling
PP	Polypropylene
RAM	Random-Access Memory
RGB	Red, Green, and Blue
SD	Standard Deviation
SSIM	Structural Similarity
UE4	Unreal Engine 4
VE	Virtual Environment
VRAM	Video Random-Access Memory

Chapter 1: Introduction

The past few decades have seen remarkable advances in creating realistic visual effects for computer animations and games. However, to enable fully immersive experience for these computer-generated realities, it is also important to incorporate the sense of hearing through sounds, which provide important functions to increase human perception of the virtual world [1], [2]. For example, environmental sounds are often used to set the atmosphere and pace of a scene, and digital audio effects are used to provide feedback to a player's actions in a video game.

In movies and television shows, sounds are commonly generated using hand-selected recordings, or performed by "Foley" artists (originally developed by radio sound artist Jack Donovan Foley, hence the name "Foley" sound) in a sound studio during post production. These artists use a variety of props and materials to recreate and record the sound effects of a scene while watching the film clip to ensure that the audio is properly synchronized with the visuals. Such an approach can produce immersive and high-quality results, but requires a large amount of manual work. The more complicated the scene, the more sounds are needed. For real-time applications such as video games, the process usually requires creating a pre-recorded sound sample database. The desired sound effects can then be triggered by computer-generated events such as an impact or explosion [3], [4].

1.1 Problem with Stored Recordings

In the world of physics, sound is a sequence of longitudinal waves propagating through compressible media such as a gas, liquid or solid. If such acoustic waves have frequency contents that lie within the human hearing range (20 Hz – 20 KHz), our brain is able to perceive them. Ever since the invention of the phonograph by Thomas Alva Edison in 1877, technologies to preserve and reproduce sound waves have evolved through multiple eras. Nowadays, the recording of sound is typically accomplished via digital encoding, where audio signals are formed by sampling analog wave signals at regular intervals in time, and then quantizing the amplitudes to discrete values [5].

A common technique to reproduce or synthesize digital sound is using stored recordings as the basic data. This data is manipulated and played back to yield the final output audio. As an example, consider a scenario where a basketball is dropped onto a floor. When the ball hits the floor, we want to synthesize the appropriate sound. A simple approach is to record a sample of an actual basketball bouncing, and load this sample in memory. When a bouncing event occurs, the sample is then sent to the audio hardware. This is called wavetable (or sampling) synthesis [6]. Filters are usually added to enhance the quality of the synthesis, allowing control of spectral brightness as a function of intensity, and to achieve more variety of sounds from a given sample.

Furthermore, for interactive applications, we also want to take the impact force into consideration to increase the immersion of the bouncing sounds. One solution would be to record a set of samples for various levels of impact. Then, at each impact event, the

sample that corresponds best with the actual force will be loaded. A lower-cost solution would be to change the amplitude of the sample in real-time before loading it, so that the volume of the sound can correspond to a particular impact force. This is a very simple example of parameterized synthesis, where in this case the force of impact is translated into audio volume. Consequently, a single prototype sound can be changed in real-time to create many other sounds.

However, such a synthesis method can sacrifice some level of realism, because we assume that the only effect of the impact force is a change in volume. This may be true in a simple approximation, but non-linear effects can cause the timbre (perceived tone quality) of the sound to change also with the impact force. Moreover, what if we drop the basketball onto a hard tile, and then to a soft carpet? The timbre of the sound should change, depending on the floor texture, because this will excite different resonance modes in different proportions. To deal with this, we can record some samples of the sounds corresponding to various impact locations, and play back the closest one, or do an interpolation between them. This is called sound morphing. It can be done in the time domain because the spectral content of the two sounds is usually the same.

Ultimately, by using numerous recordings, filters, and various interpolation methods, the synthesized sounds can achieve a high level of realism. However, in computer animations and games, the available recordings and resources are usually limited and do not exactly match the desired application and circumstance. To produce convincing, realistic and interactive sounds for a given scene or event, parameters such as timing and spectral

characteristic should all be flexible for easy manipulations. For simple effects such as footsteps, punching or breaking, such manipulations of recordings are appropriate. Whereas for more complex interactive sounds such as sliding or rolling, it can be quite labor intensive and tedious to recreate all possible sounds that would match the motion of the objects. Another major drawback with stored recordings is that the synthesized sounds are treated as afterthoughts, often added at the end of the development cycle which have little to do with the actual geometry and physics of the objects. This can create a gap between the design of an object's visual and auditory properties and in turn lead to more manual work of tweaking and experimenting with sound synthesis. Moreover, it results in a lack of variety in the sample database, which would cause the same sounds to be repeated over and over again. Therefore, for computer-mediated applications, simply using pre-recorded sounds cannot achieve the desired sonic quality and responsiveness. To generate synchronized audio content for these applications, the synthesis method should be able to automatically respond to both external (input devices) and internal (simulation) variables. As a result, sound designers have begun to utilize computer-assisted techniques such as procedural generation to address these issues.

1.2 Procedurally-Generated Audio

Procedural content generation (PCG) is a computing technique which creates data using algorithms that combine both manual content production and computer-generated randomness [4]. In computer graphics and games, it has become a popular method to automate or aid the generation of large quantities of assets such as textures, models and game levels. Similarly, the idea behind procedurally-generated audio is to produce sound

by executing a well-defined procedure based on a set of programmatic rules and live input [7]. Traditional audio synthesis has its root in recording, where real-world audio signals are captured by a microphone, mixed, processed, and then mastered into the final form. Sound effects created this way are fixed. Every time they are replayed, the form remains the same. On the contrary, procedural audio is more versatile and has the ability to be different each time it is played. It behaves like a form of unfixed recording, where the program starts running, usually with some parameters to determine the desired output, and it either immediately begins to produce sound that changes continuously over time, or responds to input by generating corresponding output sound.

To further illustrate the advantages of procedural audio over stored recordings, we list some of its key features below, which are adapted from Farnell's survey paper on procedural audio and its applications in video games [7].

- **Interactivity.** Procedural audio has the ability to modify its output in real-time during playback. Because of this, it is a very popular method among interactive sound rendering systems for dynamic virtual environments (VEs) [8]–[10].
- **Non-Linearity and Variety.** Procedural audio uses additional input to apply new sounds or edit existing ones. The order of the data is not fixed and the process from one value to the next is non-linear. It functions similarly to virtual musical instruments (such as MIDI software). The resulting sounds can be highly variable with continuous real-time parameters being applied.

- **Automation.** Procedural audio allows sounds to be automatically generated at runtime. This liberates the sound designers from exhaustively considering every possible drop, break or impact sound. Instead, they can now concentrate more on tuning the aesthetic of critical sounds.
- **Object-Based.** The sound designer's focus switches from specific instances of sounds to the behavior and physics of the whole classes of objects. They now work more like programmers, creating and modifying sound objects. This brings the sound and 3D object designer to collaborate closer together, thus closing the gap between the objects' visual and audio properties.
- **Adaptable Cost.** Procedural audio has an adaptable cost for computation. The more complex the sound is, the more computing power it needs. Such dynamic cost has a great benefit. To reduce resource consumption, we can use different level of details to filter out distant or irrelevant sounds. Moreover, procedural audio does not need much space to store. It can be treated like a program, where we only need to store a few lines of codes and the intended results will be generated on the go.

1.3 Problem Statement

Over the last few decades, there has been an increasing interest in integrating procedural audio generation with computer graphics and interactive applications. One particularly interesting topic is the synthesis of sound for rigid/soft-body interactions. Most of the rigid-body materials (such as metal, glass, or ceramic) can be assumed to be homogeneous and isotropic, which means their compositions and properties are uniform throughout the entire material, and when measured across all directions, the properties of

materials (such as Young's modulus) are the same. Therefore, rigid bodies will typically exhibit sinusoidal oscillations with strong ringing resonances under an external impulse, and their sounds are relatively easy to model using linear physics-based synthesis [6].

However, for soft-bodies (such as cloth, rope, or jelly), such assumption does not hold as their materials commonly consist of layers of highly complicated compositions, which are distinctively heterogeneous and anisotropic when measured along different directions. Thus, their sounds tend to be noisy, yet have a very natural and organic characteristic, which can be quite attuned to the presence of digital synthesis artifacts. With such highly complex structures, direct numerical simulation of physics-based acoustic emissions from soft bodies can be extremely complicated. Existing literatures are either limited to simulating the underlying physical principles of certain interactions that only address a small number of sound phenomena or rely on data-driven synthesis that utilizes recorded samples. Therefore, it is challenging to produce sound effects that can precisely capture the complex interactions occurring with soft bodies' deformation and contact state. We also need to consider the computational cost, as soft-body dynamics themselves would consume a large amount of computing power.

In this thesis, we address the problem of how to generate realistic sounds for soft-body interactions, both automatically and efficiently, in computer animations and games. Our goal is to develop practical methods that do not require computationally expensive physics simulations and can be integrated with most of the commercial off-the-shelf soft-body dynamics solvers. Instead of computing sound emission and propagation directly,

we propose to synthesize audio by utilizing the characteristics of real-world recordings. Our data-driven methods promise fast and automatic generation of plausible sounds that fully correspond to the actual motion of most common soft bodies.

1.4 Research Scope and Objective

The scope of soft-body interactions in the field of computer graphics and gaming is quite broad. With the advancement of numerous computer simulation techniques in recent years, the dynamics of complex deformable models have expanded tremendously in many key areas such as hair physics, cloth animations, and fluids, to name a few [11], [12]. To narrow down the scope of this research, we mainly focus on the types of interactions that can be simulated from commercially available soft-body dynamics solvers. Most of these simulators (such as *Bullet*, *Havok*, *Maya nCloth*/*nParticle*, *NVIDIA FleX*, and *Unreal Engine*) provide particle-based systems to save computational power, where the meshes of the objects are converted into particles (or nodes) connected by tension and constraints [13]. These systems may not be necessarily accurate, but can provide visually plausible estimations with high computational efficiency.

Generally, two types of soft-body interactions can be simulated using these types of systems: elastic deformations and plastic deformation [14], [15]. The first type of interactions usually involve small deformations where the objects tend to completely or closely recover their original shapes after the external impulses (or stress field) are removed. Common examples include ball bouncing, skin interactions, and jelly wobbling. Whereas the second type of interactions deal with large deformations that are irreversible

even after the external forces or stresses are removed. Such interactions are normally seen in cloth and rope simulations, where the shapes of the objects change permanently.

To investigate the sound generated from the two types of soft-body interactions mentioned above, we carry out research that addresses three interrelated topics in three phases:

- For phase one, we aim to synthesize sounds for elastic deformations, where we consider two types of contact models with six common interaction scenarios: discrete contact - bouncing a basketball, hand clapping, dropping a jelly; and continuous contact – hand rubbing, apple slicing, shaking jelly on a plate. A data-driven, granular synthesis based method is proposed, which is suitable for off-line animations.
- Then, in phase two, we look at the sound from plastic deformations, particularly cloth and rope interactions as they are most widely used in many interactive applications. We present an improved real-time method based on concatenative synthesis that is specifically tailored for cloth and rope simulations.
- Finally, phase three evaluates our synthesized results, where we conduct a user study to examine the perceptual quality of the sounds compared to recorded ones, as well as propose a deep learning based objective metric to measure the similarity between our results and real recordings.

1.5 Contributions

Our work presents one of the first efforts to develop practical synthesis methods for soft-body interaction sounds. In general, our synthesis methods can save a large amount of manual work over traditional Foley artistry with stored recordings, and are much more computationally efficient compared to sound synthesis built upon physical principles. The key contributions of this thesis are as follows:

- A novel data-driven method based on granular synthesis is proposed to automatically generate the sounds from elastic deformations.
- Two contact sound models (impulsive and continuous) are developed and integrated in *Autodesk Maya*, which are able to produce audio for a variety of off-line soft-body animations consisting of small deformations.
- An improved motion-driven and concatenative synthesis method is presented to produce the audio for two of the most common plastic deformations – cloth and rope interactions.
- We develop a real-time shape-dependent parameter estimation model using the *Unreal Engine* that can effectively analyze sound-related deformations and contact states of cloth (friction and crumpling) and rope objects (friction and aerodynamics).
- A detailed source database acquisition process is described, and a fast feature correspondence method is introduced to index these database sound clips to correspond to the fast motions of the sounding objects.
- A subjective evaluation criterion consisting of a three-part perceptual study is developed based on previous protocols, which can effectively assess the

recognisability, quality, and synchronization of the synthesized sounds, and demonstrate that they compare favorably to their corresponding recordings.

- Two new objective metrics based on deep learning techniques are also investigated to measure the recognisability and quality of the synthesized sounds from a different angle, and the results show the pros and cons of these metrics compared to the subjective methods.

1.6 Thesis Organization

This thesis leads to a series of published (or awaiting to be published) works [16]–[18]. Please refer to Section 1.7 for a list of our publications. Specifically, paper #3 (shown in Section 1.7) is built on Chapters 1 and 2, and paper #4, #2 and #1 are based on Chapters 3, 4 and 5, respectively.

The remainder of this thesis is organized as follows: In Chapter 2, we survey the state-of-the-art of procedurally-generated audio in the computer graphics and gaming communities, including both rigid/soft bodies and relative evaluation methods. We then discuss the benefits and drawbacks of these methods for generating soft-body sounds, as well as how we aim to improve them in this thesis.

In Chapter 3, we describe the generation of audio for elastically deformable models (research phase 1). An off-line data-driven synthesis method is proposed that is able to retarget a database of pre-recorded sound tracks according to the motion of input

animations. This method is demonstrated on a variety of elastic deformations, which include basketball bouncing, apple slicing, hand clapping, and a jelly simulation.

In Chapter 4, we introduce an improved method to synthesize the sound of plastically deformable models in real time (research phase 2). For this type of interactions, we select two types of most commonly used models – cloth and rope, and use a fast geometric analysis technique to calculate audible motion events at run time. This system is implemented in a game engine, and demonstrated with a variety of examples.

Chapter 5 presents the evaluation results of our synthesized sounds using both subjective and objective criteria (research phase 3). The subjective evaluation includes a three part user study to assess the perceptual effectiveness of the above methods, while the objective metrics measure the similarities between the output sounds and real recordings. We also compare these two evaluation methods and provide some insights.

Finally, in Chapter 6, we conclude this thesis and outline directions for future research.

1.7 List of Author's Publications

1. F. Su and C. Joslin, "Subjective and Objective Evaluation of Procedurally-Generated Audio for Soft-Body Interactions," under review in *Journal of the Audio Engineering Society*, 2020.

2. F. Su and C. Joslin, “Procedural Sound Generation for Soft Bodies in Video Games,” in *ACM SIGGRAPH Conference on Motion, Interaction and Games 2019*, pp. 1–12, 2019. https://doi.org/10.1145/3359566.3360068
3. F. Su and C. Joslin, “Toward Generating Realistic Sounds for Soft Bodies : A Review,” in *ACM Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, pp. 199–206, 2019. https://doi.org/10.1145/3356590.3356620
4. F. Su and C. Joslin, “Procedurally-Generated Audio for Soft-Body Animations,” in *ACM Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, pp. 1–8, 2018. https://doi.org/10.1145/3243274.3243285

Chapter 2: Related Works

Physically-Based Sound Synthesis	Rigid Body: [19]-[32], [37]-[40], [70], [94]
	Fracture Solid: [33]; Elastic Rod: [48]
	Thin-Shell: [34]-[36]
	Fluid: [41]-[45]
	Aerodynamic: [46], [47]
Data-Driven Sound Synthesis	Rigid Body: [49], [51], [54]
	Soft Body: [52]
Hybrid Sound Synthesis	Rigid Body: [57]-[60]
	Combustion: [55]

Figure 1 Summary of previous related works on procedural audio generation in computer animations and games.

In this chapter, we summarize previous works on procedural sound generation for both rigid and soft bodies in computer animations and interactive applications (Figure 1). We begin with physically-based approaches that produce sounds by simulating the surface vibrations of elastic objects under external influences (see Appendix A.1 for detail), followed by data-driven approaches, which focus on using or reproducing characteristics of recorded data when the underlying physical systems are too expensive to simulate (see Appendix A.2–A.6 for detail). We then look at some hybrid methods that can estimate parameters of physically-based synthesis using real world recordings. Next, we review the associated subjective evaluation, and introduce important concepts and related works

for objective sound evaluation. Finally, we compare the pros and cons of these methods, and identify their problems for synthesizing soft-body audio.

2.1 Physically-Based Approach

2.1.1 Linear Modal Synthesis for Rigid Bodies

Procedural audio has a long history in rigid-body sound synthesis. Since the 1990s, there exist a number of prior works on modeling the elastic vibration and acoustic radiation from animated solids using linear modal analysis (LMA, see Appendix A.1). Van den Doel et al. [19]–[21] have introduced the concept of using linear modal oscillators to generate contact sounds for rigid bodies with simple shapes. Later, O'Brien et al. [22], [23] described a finite element method (FEM) to model the vibrations of nonlinear arbitrarily-shaped objects at audio rates. Moreover, to account for sound radiations from linear modal vibration models, James et al. [24] introduced pre-computed acoustic transfer (PAT) to provide real-time estimation of frequency-domain radiation fields that satisfied the Helmholtz wave equation (a.4 in Appendix A.1). Later on, Li et al. [25] improved this method via a more efficient and general pre-computation, which eased the parameter tuning for desired sounds and increased the runtime efficiency. More recently, Wang and James [26] proposed a time-domain PAT that used optimal mode conflation and GPU acceleration to greatly reduce the preprocessing costs. In general, although LMA has the advantage that the simulation parameters correspond directly to physically measurable quantities and the results are accurate, the main drawback is the complexity of implementation and low speed of simulation.

To accelerate the simulation speed of modal synthesis, several simplified modal sound models have been proposed. For example, timbre trees [27], mass-spring systems [28], three-level surface representations [29], perceptual clustering [30], frequency-domain modal models [31], eigenmode compression [32], and pre-computed soundbanks [33]. These methods all have the potential for real-time rigid-body sound synthesis, but would always require some speed-accuracy trade-offs.

2.1.2 Non-Linear Thin Shells

Linear modal sound models can provide convincing physically-based sound sources for rigid bodies. Unfortunately, they are not very effective when it comes to thin-shell objects, such as sheet metals or plastic containers. Chadwick et al. [34] proposed an efficient far-field acoustic transfer (FFAT) model to synthesize realistic sounds from thin-shell structures undergoing small but non-linear vibrations. This method was extended by Cirio et al. [35], where they focused on synthesizing crumpling sound (involving gross elastic and plastic deformation) for thin shells. Recently, Cirio et al. [36] presented a multi-scale reduced-order method to accelerate the simulation process. They also enriched the simulated sounds with wave turbulence. Although these methods are able to produce synthetic high-quality thin-shell sounds, they cannot be used to simulate large deformations due to mode locking.

2.1.3 High-Quality Modal Sound

While modal sound models are widely used for rigid bodies, there still remain a number of significant contact-related deficiencies that limit the realism of the resulting contact

sounds. To resolve modal vibration in both collision and frictional contact sounds, Zheng and James [37] proposed an adaptive modal synthesis for contact resolution that could synthesize a range of challenging sound phenomena. To add richness to impact sounds, Chadwick et al. [38], [39] introduced pre-computed acceleration noise (PAN), which accounted for the sound produced when a small object experienced abrupt rigid-body acceleration due to collisions or other contact events. Moreover, Wang et al. [40] explored high-quality offline wave-based sound synthesis for animation-driven sound radiation using a finite-difference time-domain (FDTD) wavesolver. Although the simulation costs were extremely expensive (for example, a 10-second metal sheet shaking would take 24 hours with 36 CPU cores), a rich variety of high-fidelity sound effects could be generated.

2.1.4 More Complex Animation-Sound Phenomena

Beyond rigid-bodies, there are a number of recent works focusing on simulating more complicated animation-sound phenomena. Van den Doel [41] introduced the first method in computer graphics to generate fluid sounds based on a bubble vibration model. Similar methods were also considered by Zheng and James [42] and Moss et al. [43] for different types of fluid simulators. Later on, Langlois et al. [44] proposed an improved two-phase incompressible fluid-air model to synthesize bubble sounds, which also led to enhanced visual fidelity of fluids in computer animation. Finally, Liu et al. [45] simulated the sound of raindrops using an efficient physics-based statistical model.

Besides fluids, Dobashi et al. [46], [47] devised a method to efficiently render vortex-based aerodynamic sounds. Plausible sound effects such as swinging clubs and whistling wind could be generated with this method. Moreover, Zheng and James [33] synthesized sounds from brittle fracture solids using LMA. Schweickart et al. [48] presented a method to generate physically-based sound for elastic rods. Their approach used a pre-computed dipole model to resolve sound radiation, and supported highly deformable objects such as a slinky. However, these methods currently target limited classes of sounds and cannot render convincing audio for large elastic and plastic deformations. To the best of our knowledge, no prior work has addressed physically-based sound generation directly from the dynamics of soft-body interactions.

2.2 Data-Driven Approach

2.2.1 Motion-Driven Synthesis

The main limitation of physically-based approaches is that they require large amounts of computation (or pre-computation) and parameter tuning. To avoid such expensive simulations, many works take a different approach that focuses on utilizing characteristics of recorded data. For example, Cardle et al. [49] resynthesized sounds for new input 2D animations based on existing soundtracks. They used audio segments from source animations to train a statistical model that could generate variants of the original audio to match the target animations. New sounds could be automatically produced at a coarser level without using any physical models of the motions.

2.2.2 Granular Synthesis

Another popular data-driven method is granular synthesis [50], which manipulates short grains of sounds (brief micro-acoustic events, typically 1-100 ms, containing meaningful waveforms) to form new audio sequences. This method has been adapted to synthesize audio for rigid-body simulators [51], where audio grains were retargeted from a database of recordings to best match the extracted motion events of the physics engines/input animations. Plausible breaking, sliding and rolling sounds could be generated. However, no visual or acoustic simulations of 3D interaction processes were considered, and the pre-processing time required to calculate the correlation pattern took four hours, making it impractical for real-time synthesis.

2.2.3 Concatenative Sound Synthesis

Inspired by the above synthesis techniques, An et al. [52] addressed the issue of synthesizing cloth sounds based on a concatenative sound synthesis (CSS) method [53]. First, they analyzed the motions of an input cloth animation and generated a crumpling and a friction sound model that were used to synthesize a low-quality target sound signal. Secondly, they used a hand-tuned warping function to select a sequence of pre-recorded microsound units that best matched the target signal. These selected units were then concatenated together to produce the final sound. Similarly, Schreck et al. [54] adapted this method to paper material with real-time processing ability. Although these highly specialized techniques can avoid direct numerical simulations, they are only effective for certain objects and cannot be generalized to other types of interactions. Furthermore,

most of the abovementioned methods require lengthy pre-computations with a lot of manual efforts, which results in higher computational cost.

2.3 Hybrid Approach

2.3.1 Combustion Sound

While both physically-based and data-driven syntheses have their own pros and cons, sometimes it would be more beneficial to combine these two. Chadwick and James [55] presented a hybrid approach to synthesize plausible sounds for combustion phenomena. They first synthesized a low-frequency flame sound using a physically-based combustion sound model driven with data from a visual flame simulation. Next, they proposed to use spectral bandwidth extension and data-driven texture synthesis based on real flame recordings to synthesize additional high-frequency flame sound content. Various plausible combustion sounds could be generated, from small candle flames to large flame jets.

2.3.2 Parameter Estimation

One of the key challenges for physically-based LMA is how to determine satisfactory parameters that can recreate realistic audio. Real objects can have complex shapes and material composition, hidden internal structure, and subtle boundary conditions [56]. Therefore, it is difficult to directly measure these parameters for constructing the required sound model. Richmond and Pai [57], [58] proposed a robotic measurement of contact sound responses of real-world objects. Their robotic system was able to acquire sound models from existing objects by recording sound samples produced by striking the

objects at precisely registered surface locations. With this system, they could collect a large number of registered sound samples automatically. These samples were then used to estimate physics parameters of an impulse response sound model.

Moreover, Lloyd et al. [59] utilized recorded data to estimate modal parameters for impact sounds in video games. Instead of using ideal exponential decay, they applied actual amplitude envelopes extracted from recorded clips to achieve higher quality model synthesis. Additionally, they computed a residual to capture details of the sound missing in the modal representation. More recently, Ren et al. [60] designed a novel parameter estimation algorithm using pre-recorded audio clips to search the best material parameters for modal synthesis. Similar to Lloyd et al. [59], their method also compensated the differences between real-world recordings and modal-synthesized sounds. The resulting sounds well preserved the same sense of material as recorded ones. However, as with all the other modal synthesized sounds, this feature extraction and parameter estimation process assumes that the recorded material is homogeneous and isotropic, which is not the case for most of the soft-body objects. For example, composites such as rubber and cloth are highly anisotropic when measured along any direction of their surfaces. The anisotropy affects the sound quality significantly. Thus, it is practically impossible to estimate the material parameters for soft-bodies and generate their modal synthesized sounds.

2.4 Evaluation Methods for Procedurally-Generated Audio

Evaluation for procedural audio is commonly done by experienced sound designers, or based on psychoacoustics studies, where a number of human listeners are recruited to judge the auditory quality of samples with their mean opinion scores (MOSs) [61]. Such subjective evaluation is usually very effective, but can be quite time-consuming, and would require expert listeners. Careful considerations must also be taken to avoid bias in auditory perception, which may lead to less accurate results. In contrast, there are times where objective evaluation needs to be held, especially when we are measuring the quantitative properties of the generated sounds. Typically, objective evaluation is in the form of performance metric, such as error rate or signal-to-noise ratio [62]. This kind of metric is fast, reliable and accurate. However, it cannot be directly used to measure the perceptual quality of the audio.

To solve this issue, a common approach is to compare the synthesized audio to its corresponding recording, and compute a similarity score between the two. This type of task can be categorized as audio fingerprinting in the music community, where the aim is to derive a compact representation of the audio that can be efficiently matched against other audio [63]. A popular application of audio fingerprinting is query-by-example music recognition program such as *Shazam* [64]. It recognizes music by matching the extracted hash tokens of the sample with the existing database. However, music is strongly structured and clearly demarcated [65], whereas soft-body sounds, like other environmental sounds, have no common structures. This poses a number of unique

challenges, and would require a more advanced feature extraction algorithm that can precisely capture the characteristics of soft-body audio.

Given the recent advances of deep learning technologies (a type of machine learning that uses artificial neural networks to learn from large amounts of data) for audio signal processing, it is possible to find an adequate solution since deep neural networks have proven to be effective for handling large amounts of data and modeling complex features with increased computing power [66]. These learned feature representations can be used for audio classification and recognition tasks [67]. Furthermore, the success of generative adversarial networks (GANs) in generating images [68] has also enabled the possibility of recreating temporal structures for audio signals, which leads to the development of new quantitative evaluation methods for these generative models. Such metrics were found to correlate well with subjective evaluation [69], thus would be a promising candidate to evaluate more complex soft-body sounds.

2.4.1 Subjective Evaluation

Normally, validation of rigid-body sound is done by comparing the frequency spectra of the simulated output and recorded data [70]. As soft bodies tend to produce noise-like audio with complicated structures, direct comparison between simulated and recorded spectrograms cannot convey as much information as for rigid bodies in terms of structural distribution of acoustic events and their frequency ranges. To better understand the auditory perception of procedurally-generated sound, a number of other works included perceptual user studies to evaluate their results [25], [29], [35], [43], [45], [60],

[61]. Unfortunately, none of these studies has ever been conducted on the sound of soft bodies. Therefore, to fill this gap, we aim to use several of these subjective evaluation methods to assess important properties of our generated soft-body audio.

2.4.2 Objective Evaluation

The recent surge in deep learning has enabled the development of many practical applications in the area of audio signal processing. Audio signals are commonly converted into sequential time-frequency representations for processing [62]. To increase efficiency, many researchers have started to adapt the algorithms from image processing to solve similar problems in audio. This process usually involves transforming an audio file into a series of images, and then using a neural network to process them. One of the most widely-used deep neural networks is convolutional neural network (CNN) [71], which is designed specifically to analyze the variability of two-dimensional images. Over the past several years, a large amount of researchers have shown that CNNs can be successfully trained to deal with spectral images of sounds for many applications such as speech recognition [72], acoustic scene classification [73]–[76], and acoustic event detection [77]–[81].

Moreover, with the advent of larger datasets such as *ImageNet* [82] as well as deeper CNN architectures such as *AlexNet* [83] and *Inception* [84], CNNs have also seen wide success in deep generative models such as GANs [68], which learn to produce realistic samples of a given dataset from low-dimensional, random latent vectors. GANs are powerful tools for generating images, and are also useful in the audio domain for source

separation, musical timbre transfer, and speech enhancement [62]. More recently, Donahue et al. [85] attempted to apply GANs to unsupervised synthesis of audio based on raw-waveform (WaveGAN) and spectrogram (SpecGAN). They demonstrated that both strategies were capable of synthesizing audio signals with global structures, suitable for sound effect and speech generation.

Despite the impressive results that GANs have produced in recent years, evaluation of these outputs remains a difficult problem. As a result, a variety of objective evaluation criteria have been developed [86], among which the Inception Score (IS) [69] and Fréchet Inception Distance (FID) [87] are the most widely used metrics for images. Both methods apply a pre-trained CNN (*Inception v3*) to generated images and calculate statistics of the network's output or at an intermediate layer. Typically, IS seeks to capture two important properties of the generated images: recognisability and diversity. However, it does not utilize the actual statistics of the samples themselves. To improve the IS, FID uses the neural network to extract features of synthesized images and compare them to the extracted features of real ones. Both metrics are reported to be consistent with human judgment, and have shown promise for evaluating procedural sound [85].

The important difference between a generative sound model and procedural sound generation is that a generative model is an unsupervised strategy for mapping low-dimensional latent vectors to high-dimensional audio data, whereas procedural generation is largely a supervised approach that creates audio data through a combination of human effort and computer algorithm. Despite this difference, however, both methods are able to

produce plausible results that preserve characteristics of the original sound database. Therefore, it would be intuitive to apply similar evaluation metrics from generative models to procedural audio. In Chapter 5 of this thesis, we investigate two of these objective evaluation criteria (namely IS and FID) for our synthesized soft-body sounds, and also verify their correlations with subjective methods.

Table 1 Summary of pros and cons of different methods for procedurally-generated audio.

Methods	Pros	Cons
Modal Synthesis (Section 2.1.1, Appendix A.1)	• Physics-based • Fully automatic • Accurate	• Not ideal for large deformation • Computationally expensive • Complex parameter tuning • Memory-inefficient
Motion-Driven Synthesis (Section 2.2.1, Appendix A.3)	• Natural design flow • Automatic • Easy to use	• Limited to noise-like sound textures • Does not work for progressive sounds • Low quality
Granular Synthesis (Section 2.2.2, Appendix A.5)	• Ideal for representing timbre evolution as well as stochastic sounds • More ways to organize sound textures	• Not effective for smooth results • Offers no control and analysis method
CSS (Section 2.2.3, Appendix A.6)	• More efficient reuse of source sounds • High level control • More flexible	• Requires a large amount of database recordings

2.5 Summary

Table 1 summarizes the pros and cons of all the methods mentioned above for procedural sound generation. Please refer to Appendix A for a detailed description of each method.

We see that linear modal synthesis promises fully automatic sound generation with high accuracy, and has been widely considered in simulating rigid-body sounds. However, this method currently cannot render convincing soft-body sounds due to the complexity of modeling large-deformation vibrations (see Appendix A.1 for more detail). Direct numerical simulation of physically-based acoustic radiations from deformable bodies would also suffer from large memory requirements and lead to slower computation time far beyond traditional soft-body simulations. Furthermore, soft bodies are usually composed of many layers of materials (such as skin or cloth), each with distinctive mechanical and acoustic properties. This would make it even more complicated to estimate and tune the modal parameters.

On the other hand, data-driven sound synthesis is introduced as a means of synthesizing long audio streams from short example audio clips, which is relatively cheap to compute (see Appendix A.2 for detail). Therefore, for practical applications such as games or computer-animated movies, it would be preferable to use data-driven approaches based on pre-recorded sound effects. Although most of these methods could, in principle, be modified to generate soft-body sound textures, the challenge here, however, is how these textures can be controlled to create meaningful sounds that correspond to the complex motions/interactions of soft bodies.

While several data-driven methods have been adapted to synthesize familiar sounds for many computer-simulated objects (Section 2.2), there still remain a number of gaps when it comes to generate the sounds for soft bodies. Motion-driven synthesis (Appendix A.3)

is, to date, the mostly used method for synthesizing procedural audio [49], [52]. Although this approach allows for automatic sound generation based on the input animations, the quality of the results are generally considered quite low.

Granular synthesis (Appendix A.5) provides more ways of combining short sound samples to form new sequences. It is a classic method for resynthesizing micro-sound details, and is ideal to produce stochastic and noise-like sounds, which would be desirable for soft-body interactions. However, the method itself is a generative technique that lacks analysis methods and meaningful controls. To resynthesize micro sounds that correspond to the motions of soft bodies, we must also develop relevant control/analysis techniques.

Last but not least, CSS (Appendix A.6) offers more efficient and accurate input control to select and resynthesize sound units, but requires more manual work for database construction. For soft-body objects with complex structures, such database may become difficult to obtain as we need to carefully decompose each acoustic event into meaningful sound units that can match certain motion/deformation. This would require us to have a solid understanding of the characteristics of each different type of soft-body interaction.

In this thesis, we aim to fill these research gaps by designing and implementing systems that can automatically generate plausible sounds for soft-body interactions including both elastic and plastic deformations. Our methods do not require expensive physics simulations, and have improved previous data-driven synthesis by introducing more

accurate analysis and control techniques, as well as significantly reducing the computation/pre-computation time. Furthermore, we also present one of the first efforts to evaluate procedural soft-body audio using both subjective and objective approaches.

Chapter 3: Sound Synthesis for Elastic Deformation

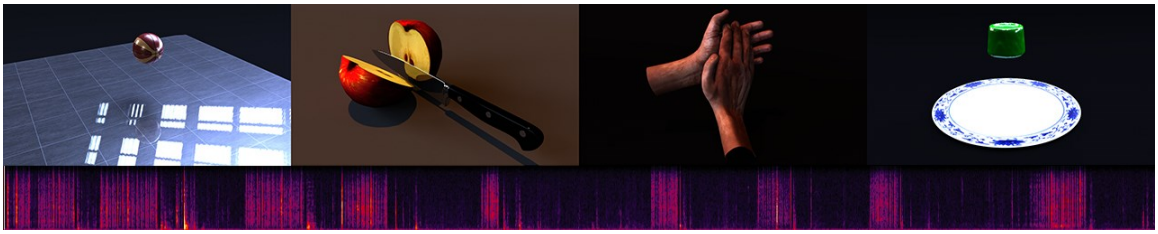


Figure 2 Common soft-body interactions with elastic deformations: a basketball bouncing, slicing a fruit, hand clapping, and jelly dropping.

3.1 Introduction

As discussed earlier, most of the soft-body interactions simulated by particle-based systems will incur two types of deformations – elastic and plastic. Elastic deformations are temporary, and are commonly seen in both rigid and soft-body simulators (Figure 2). Typically, sound will be produced from vibrations caused by these deformations. However, as stated in Section 2.5, such vibrations are too complex to be simulated with current soft-body dynamics. To date, no prior work exists that can generate sounds for this kind of deformation.

Inspired by previous data-driven approaches, we propose to synthesize audio tracks for a range of elastically deformable models using a method based on granular synthesis. An overview of our system is given in Figure 3. First, we construct a database of pre-recorded audio samples that are segmented into both continuous and impulsive sound grains. These short grains preserve important temporal and frequency information of specific types of sounding objects. Second, we introduce a novel analysis technique,

which can automatically and effectively analyze the elastic deformations of a given soft body, and extract the corresponding continuous or discrete contact model that will later be used to drive the synthesis of our audio database. Next, we label each sound grain from the database, and use an efficient mapping technique to automatically retarget both continuous and impulsive grains according to the motion parameters of our extracted contact models. Finally, we resynthesize all the selected sound grains and filter the results to generate the final audio.

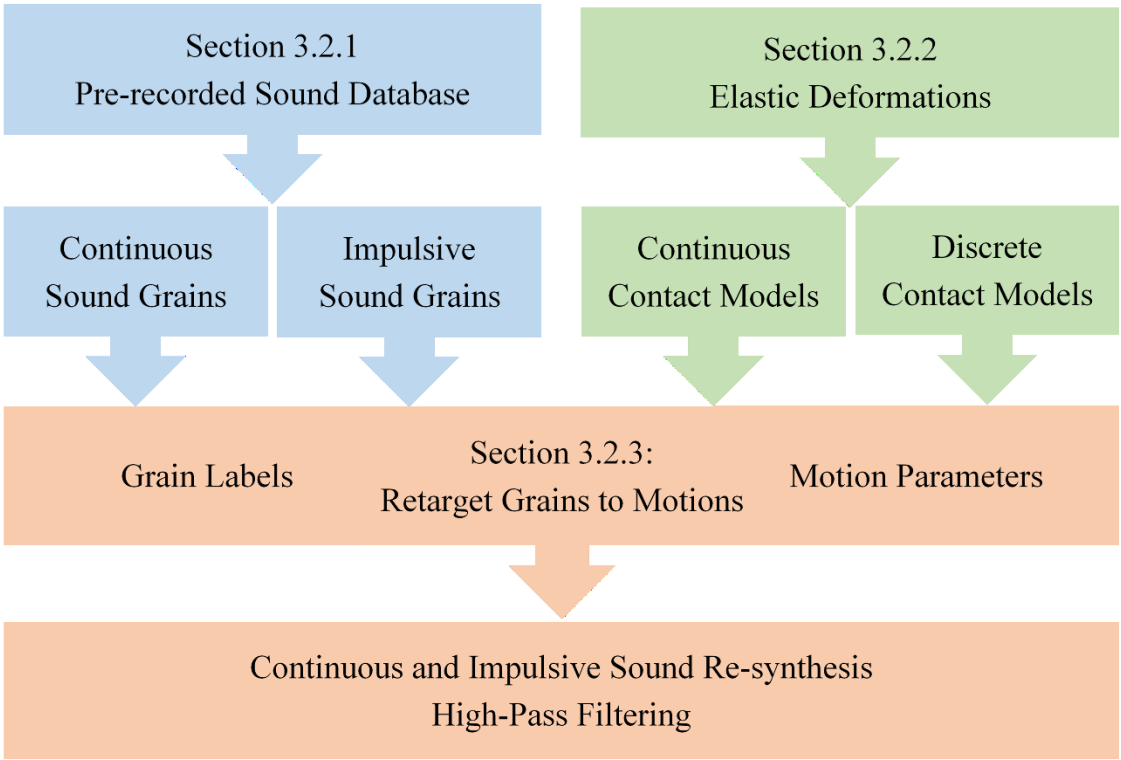


Figure 3 Overview of our data-driven synthesis approach for elastically deformable models.

Our data-driven approach improves upon the granular synthesis method by introducing a new analysis and control technique that is specifically designed to reconstruct the sound

textures from elastic deformations. It is able to produce compelling and plausible soundtracks that are automatically synchronized with the deformations and motions of the soft bodies. Our method also provides some parameters for sound designers to further tune the audio according to their needs. The memory usage and computational cost are significantly lower compared to previous granular synthesis based approach by Picard et al. [51]. The provided examples demonstrate that it is capable of simulating various elastic deformation sounds including challenging ones such as jelly dropping/wobbling.

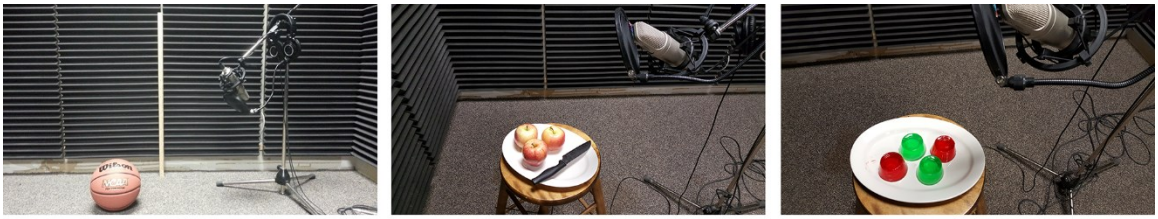
3.2 Methodology

In this section, we detail the process of our sound database collection, contact model extraction and sound synthesis algorithm.

3.2.1 Sound Database

Normally, there are two types of sounds generated from the elastic deformations of soft bodies – impulsive and continuous. Impulsive sounds are usually produced from rapid impact such as hitting, whereas continuous sounds generally include a steady-state part caused by micro collisions (a series of small collision impulses) such as sliding and rolling. Based on these phenomena, we constructed our sound databases with four unique soft-body objects using real-world recordings. For each type of audio, we chose three different interactions: for impulsive sound, we chose a basketball bouncing (hitting), hand clapping (impact with aerodynamic), and jelly dropping (large deformation); for continuous sound, we chose apple slicing (fracturing), hand rubbing (sliding), and jelly shaking (sliding and rolling). These scenarios are widely seen in many computer graphics

and gaming applications [12], and all include complex soft-body sound that would be impossible to model using a physically-based approach. For a cleaner and more detailed database, we recorded each session in an anechoic chamber using one ultra-low-noise cardioid condenser microphone (*RØDE NT2-A*, with 20 Hz – 20 KHz frequency range) mounted on a fixed location, and a 192 KHz/24 Bit analogue to digital audio interface (*PreSonus Studio 192*).



(a) (b) (c)

Figure 4 Experimental setup. (a) Basketball bouncing, (b) apple slicing, (c) jelly dropping/shaking.

Basketball Bouncing (Figure 4 (a)): We recorded the bouncing sound of a basketball on floors with different textures – a soft carpet surface and a hard tile surface. To obtain measurements of bouncing sounds versus impact energy, we dropped the basketball at various heights (0.5 m, 1.0 m, 1.5 m, and 2.0 m) with an initial velocity of 0 m/s. The ball was inflated to an air pressure of 55.158 KPa, with a circumference of 0.75 m (official size 7).

Apple Slicing (Figure 4 (b)): Cutting/slicing includes another interesting elastic deformation (which happens before and after the plastic deformation) that produces a noisy, yet very natural and organic sound. We recorded three sequences of apple slicing sounds – one at a slow speed (roughly 1 s for each cut), one at a normal speed (roughly

0.5 s for each cut), and one at a fast speed (roughly 0.2 s for each cut). For each sequence, we asked our participant to slice an apple to pieces at a relatively constant speed. This provided us with a direct map from slicing speed to the corresponding sounds created.

Jelly Dropping/Shaking (Figure 4 (c)): To create a more challenging interaction, we setup a jelly test which involves both elastic and plastic deformations. We dropped a truncated-cone-like jelly onto a ceramic plate at various heights (5 cm, 10 cm, 15 cm, 20 cm, and 30 cm) with an initial velocity of 0 m/s and recorded the impact sounds. We also performed a shaking motion where we held the plate with the jelly on it and shook the plate left-and-right at varying speed (approximately 2 to 10 shakes per second) to create large deformations on the jelly. This consistently produced many audible wobbling sounds. We recorded typically 30 seconds for each sequence.

Hand Clapping/Rubbing (Figure 5): Acoustically, a hand clap corresponds to a rapid formation and excitation of a cavity between two hands [88]. Inspired by the study of Repp [89], which indicated that the configuration of hands had important impact on spectral features of various clapping styles, we decided to include hand configuration as one of the controlled variables. Following Repp's suggestions, we used six different clapping modes (Figure 5) – modes P1-P3 kept the hands parallel and flat but changed their vertical alignment, while modes A1-A3 varied alignment in a similar way, but with the hands held at an angle. For this experiment, we recruited one female participant to perform all of the above clapping modes at varying rate – from slow (approximately 1 s between each clap) to as fast as she could (approximately 0.1 s between each clap) for

around 20 seconds. Additionally, we also asked the participant to rub her hands in mode P1 at varying rate (about 0.1 s to 0.5 s for each rubbing) for about 10 seconds. This setup effectively gave us a mapping from clapping/rubbing rate to clapping/rubbing sound in different modes.

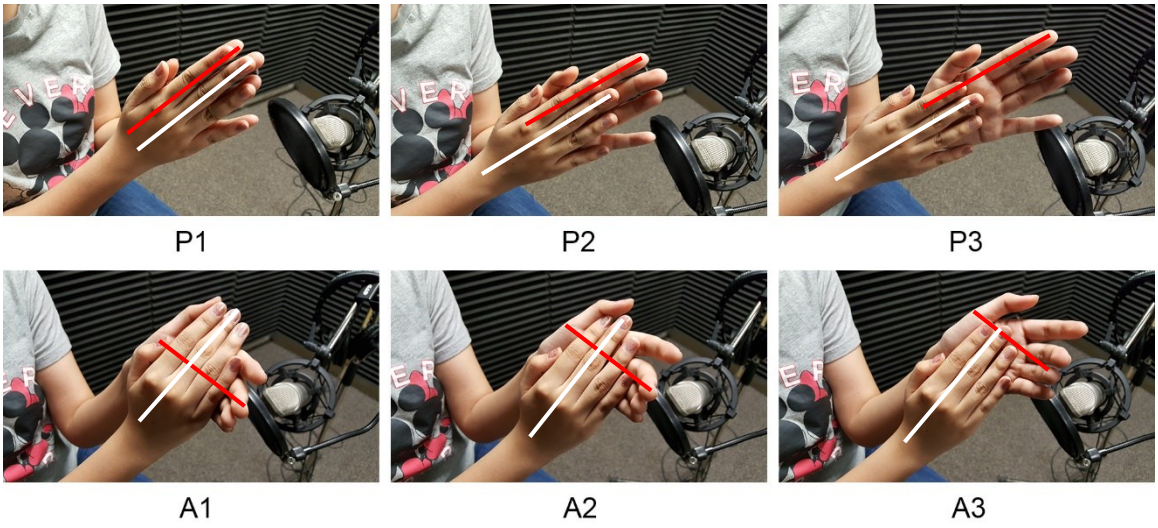


Figure 5 Six different hand clapping modes. The red and white lines here show the alignment of the left and right hands.

3.2.2 Contact Sound Models

For a convincing re-synthesis, we need to match the sound database to the contact parameters from the input animations. The first step is to extract the motion data contained in the given soft-body interactions. As contact state in common simulators (such as *Bullet* in *Maya*) is often inaccessible, we choose to estimate these contact events via position-based collision analysis at any animation frame. For simplicity, we only consider the absolute positions of each vertex.

In the case of discrete contacts (basketball bouncing, hand clapping and jelly dropping), a vertex-to-surface model (Table 2) would be sufficient to parameterize the contact events. We assume a minimum collision radius Υ which can be specified by the user (typically Υ is ranged from 0.05cm to 0.1cm), and calculate the positions of the object at each frame. If the vertex-to-surface distance is equal or less than Υ, we record a contact event i at frame t_i (i = 1,2,3,…, which indicates the index of each contact event), and calculate the relative contact energy E_i and speed s_i. For the bouncing and dropping scenarios, the contact energy can be simply estimated by the gravitational energy involved in the impact:

$$E_i = \left| mgh_{max,i} \right| \tag{3.1}$$

where m and g are respectively the object mass and gravity, and $h_{max,i}$ is the maximum vertex position on the Y axis between two consecutive contact frames (t_{i-1}, t_i). For clapping, we use the time interval Δt_c between each clap to estimate the speed s_i, namely,

$$\Delta t_{c,i} = f_s(t_i - t_{i-1})/framerate \tag{3.2}$$

Here f_s is the audio sampling rate, *framerate* is the animation frame rate, and ($t_i - t_{i-1}$) calculates the total number of frames between two consecutive contact/clapping events *i-1* and *i*. The result of the extracted motion data is a list that contains the frames where the contact will occur and the corresponding energy/speed of the object at each frame (Table 2).

In the case of continuous contacts (hand rubbing, apple slicing and jelly shaking), objects are sliding against each other. This essentially involves multiple micro-collisions at the contact area, and thus we use a different kind of model called vertex pair (Table 2). As

we observed, continuous contacts would generate steady-state collision sound textures as long as the relative velocity was not zero. Therefore, for each contacting vertex pair $vtx_{m,n}$ at each time frame, we calculate their speed $\vec{v}_m$ and $\vec{v}_n$ using a 2nd order Runge-Kutta method [90]. The animation is then segmented into several sequences when the relative velocity between each vertex pair reaches zero (i.e. $\|\vec{v}_{m|n}\| \approx 0$), and the corresponding contact frames t_i are also reported. This simplifies the re-synthesis process later as we only need to consider the retargeting of the audio grains for each segmented sequence. The reported data types for each animation are shown in Table 2.

Table 2 Contact model and resulting motion data.

Type of Interaction	**Contact Model**	**Reported Parameters**
Basketball Bouncing	Vertices: Basketball Surface: Floor (Still)	Contact Time Frame t_i Contact Energy (Gravitational E_i)
Hand Clapping	Vertices: Right Hand Surface: Left Hand (Still)	Contact Time Frame t_i Contact Speed (Clapping Interval $\Delta t_{c,i}$)
Jelly Dropping	Vertices: Jelly Surface: Plate	Contact Time Frame t_i Contact Energy (Gravitational E_i)
Hand Rubbing	Vertex Pair: Left(vtx_m)-Right (vtx_n) Hand	Time Frame t_i at 0 Relative Velocity
Apple Slicing	Vertex Pair: Apple(vtx_m)-Knife(vtx_n)	Time Frame t_i at 0 Relative Velocity
Jelly Shaking	Vertex Pair: Jelly(vtx_m)-Plate(vtx_n)	Time Frame t_i at 0 Relative Velocity

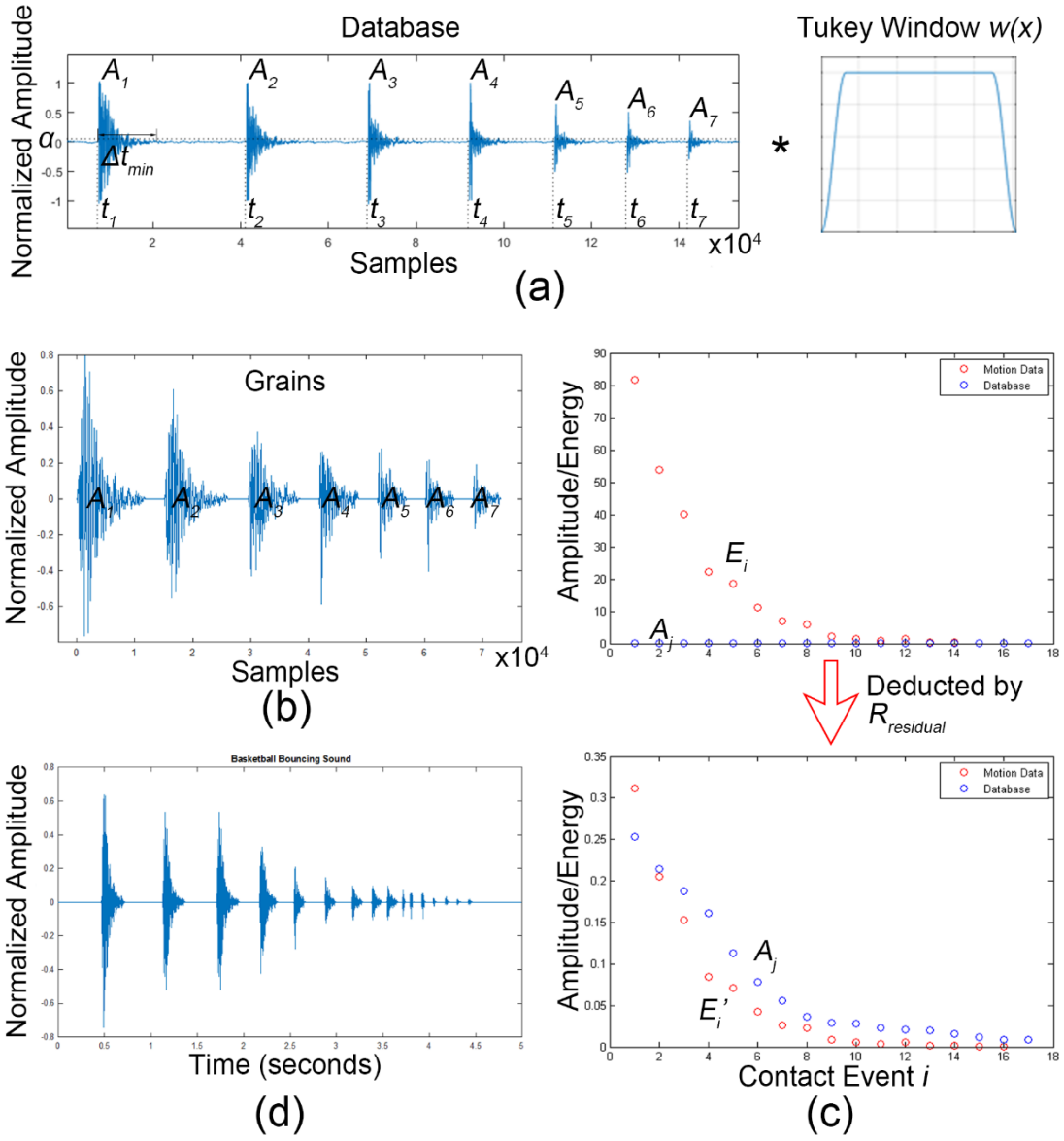


Figure 6 Synthesis of impulsive sound. (a) Peak detection and labeling, (b) grain extraction, (c) retargeting extracted grains to match the motion data, (d) re-synthesis and filtering.

3.2.3 Sound Synthesis

Once we have the sound database and motion data from the soft-body animations, we can segment these recordings into audio grains for re-synthesis. A grain of sound is a brief micro-acoustic event, with a duration near the human auditory perception, typically

between 1 and 100ms [50]. Each grain contains a waveform shaped by an amplitude envelope. A single grain serves as a building block for sound objects. By combining thousands of grains over time, we can create an apt representation of the original sound texture.

Synthesis of Impulsive Sound. As we have two types of contact models from the motion data (Table 2), we divide the source database into two categories: impulsive and continuous. Impulsive sounds are generated from discrete contacts that are easy to distinguish from each other (typically loud impact audio). For these sounds (basketball bouncing, hand clapping and jelly dropping), we develop a method that can automatically analyze the source database and extract the impulsive grains according to certain input parameters. First, to identify the local peak for each impulse signal, we look for their "attack" regions (beginning portion of the sound) by setting up an amplitude threshold $\alpha = 0.02$ (through trial and error) and a minimum time interval Δt_{min} (Figure 6 (a) Left). If the signal amplitude exceeds α, the corresponding time t_j will be marked as the beginning of an attack. We then skip over a time interval of Δt_{min} to look for the next attack. This will effectively prevent false detection of multiple attacks in the same impulse. The minimum time interval Δt_{min} can be calculated as:

$$\Delta t_{min} = \min(f_s(t_i - t_{i-1}) / framerate) \tag{3.3}$$

where f_s is the sampling rate, (t_{i-1}, t_i) are the consecutive contact frames in the motion data, and *framerate* is the animation frame rate. In the second step, we use the detected attack time t_j plus an adjustable grain length l (typically 10-100ms) to decompose the original audio into small impulsive grains. The peak value A_j is simply the local maxima

of each grain and will be used as a label (Figure 6 (a) Left). To limit the number of extracted grains, we discard the ones with low energy according to a user-defined threshold. The remaining grains are convolved with a Tukey window, giving

$$w(x) = \begin{cases} \frac{1}{2}\left\{1 + \cos\left(\frac{2\pi}{r}\left[x - \frac{r}{2}\right]\right)\right\}, & 0 \le \mathrm{x} < \frac{r}{2} \\ 1, & \frac{r}{2} \le \mathrm{x} < 1 - \frac{r}{2} \\ \frac{1}{2}\left\{1 + \cos\left(\frac{2\pi}{r}\left[x - 1 + \frac{r}{2}\right]\right)\right\}, & 1 - \frac{r}{2} \le \mathrm{x} \le 1 \end{cases} \quad (3.4)$$

with $r = 0.25$(bouncing)$/0.1$(clapping) for the ratio of taper and normalized for power conservation (Figure 6 (a) Right). This envelope provides smooth transitions at the extrema of the envelope while maximizing the effective amplitude, thus preserving the original waveform within the grain without altering its pitch too much. As the waveform can vary from grain to grain, the taper ratio r is not a fixed value. Typically, for grains with shorter duration (such as clapping), a smaller r works better.

To reconstruct the extracted audio grains (Figure 6 (b)), we use the motion data to indicate contact, and re-target grains according to the labelling of the grain obtained during the analysis. For basketball bouncing and jelly dropping, we use the reported contact energy E_i (Table 2) to search for appropriate grains. To map the grain database into the space of the motion data, the labelled peak value for each grain must be warped to create overlap with the contact energy (Figure 6 (c) Top, here the value of each grain label A_j is typically between 0-1). Therefore, we calculate the average residual between each motion data and database label value:

$$R_{residual} = \frac{1}{n}\sum_{i,j=1}^{n} E_i/A_j \tag{3.5}$$

where n is the total number of contact events, E_i is the simulated contact energy from the motion data, and A_j is the peak grain amplitude of the database. We then deduct that residual from the motion data (Figure 6 (c) Bottom):

$$E_i' = E_i/R_{residual} \tag{3.6}$$

Next, at each contact time, we simply select the nearest neighbor of the contact energy data E_i', and concatenate the selected grain i to form the new signal:

$$Grain(i) = \arg\min_j \left\| E_i' - A_j \right\|^2 \tag{3.7}$$

However, simply concatenating selected units can produce large inter-unit discontinuities which leads to undesirable low-frequency noise, thus we apply a 5th-order Butterworth high-pass filter with a 100 Hz cutoff frequency to eliminate that large discontinuities (Figure 6 (d)).

Similarly, for hand clapping sound, we label the database using the time interval $\Delta T = \{\Delta t_1, \Delta t_2, \cdots, \Delta t_{j-1}\} = \{(t_2 - t_1), (t_3 - t_2), \cdots, (t_j - t_{j-1})\}$ between each detected grain attack time t_j in the impulse signal. For each contact, we use the reported contact speed (Table 2, here we calculate the time interval $\Delta t_{c,i}$ between each clap) to select a series of candidate grains that match this interval given some minimum time metric t_{min}:

$$Grain(i) = Database(j), if \left|\Delta t_{c,i} - \Delta t_j\right| \leq t_{min} \tag{3.8}$$

To avoid repetition, we randomly pick one from all the candidates (in this case, we have 3-5 for each contact event) and concatenate all of the grains according to the contact time. The new signal is then fed to a Butterworth high-pass filter for smoothing.

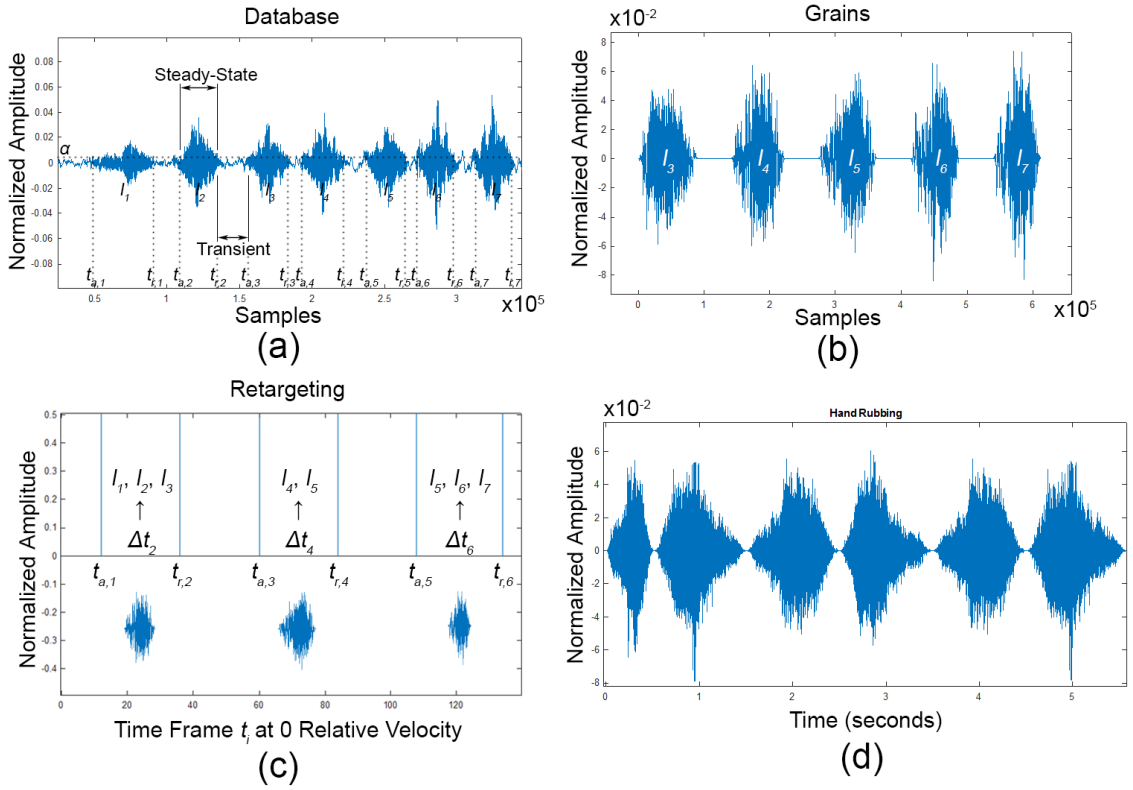


Figure 7 Synthesis of continuous sound. (a) Steady-state and transient parts, (b) grain extraction, (c) grain retargeting, (d) resampling, resynthesis, and filtering.

Synthesis of Continuous Sound. Recordings of continuous contacts contain significant steady-state and transient components. The steady-state portion consists of repetition of a periodic loop which is the characteristic of the objects during continuous contact events, while the transient part is when the relative velocity is equal or close to zero. To extract the audio grains of hand rubbing, apple slicing, and jelly shaking, we keep the steady-state portion of the continuous sound in the recordings and discard the transient part (Figure 7 (a)). Similar to the approach described in the previous section, we look for the "attack" and "release" regions ("release" here means the final decay) of each grain using

an amplitude threshold $\alpha = 0.005$ and a minimum time interval Δt_{min} to prevent false detection. This Δt_{min} can also be calculated using Equation (3.1), with F_i being the time frame at 0 relative velocity in the motion data. Once the attack and release time t_a and t_r are found, the steady-state part is simply the waveform in between, and the grain length $l_j = t_{r,j} - t_{a,j}$ can be used as a label. As a result, we obtain a dictionary of grains which preserve the key characteristic of each recordings (Figure 7 (b)).

The re-synthesis is performed by choosing from the grains of various length that best match the continuous contact time. The motion data reports the time frame t_i at zero relative contacting velocity (Table 2). These specific frames $t_{1,2,\ldots,i}$ can then be divided into multiple pairs ($t_{a,1}$, $t_{r,2}$), …, ($t_{a,i-1}$, $t_{r,i}$), with t_a denoting the beginning frame of an attack and t_r denoting the end frame after a release, so that we can retarget the steady-state grain component between these time frames. Next, we calculate the total number of samples between each reported time frame t_i (with 192 KHz sampling rate f_s and 24 FPS animation frame rate) and convert it into seconds, giving

$$\Delta t_i = f_s(t_{r,i} - t_{a,i-1})/framerate \tag{3.9}$$

The resulting Δt_i is then used to search for the best candidates in the labelled grains that match this time interval (Figure 7 (c)):

$$Grain(i) = Database(j), if\ |\Delta t_i - l_j| \leq t_{min} \tag{3.10}$$

Next, we randomly select one grain from all the candidates and resample it to fit into the interval Δt_i (Figure 7 (d)). Finally, we apply a high-pass filter to smooth out the results.

3.3 Results

We used *Autodesk Maya 2016 Extension 2* to simulate and animate all our examples. All of the animations were rendered with the *Arnold* (https://www.arnoldrenderer.com/) renderer at 24 FPS. Models and textures for the basketball, rigged hands, apple, kitchen knife and plate were obtained from *TurboSquid* (https://www.turbosquid.com/). To achieve better sound quality, we recorded and synthesized all audio clips at 192 KHz and 32 bits/sample. The analysis of the database recordings required for the grain extraction and labeling was performed in an off-line process using *MATLAB R2017b*. Our contact models were implemented in *Maya* using a plugin we wrote in *Python*. This plugin could directly analyze the deformations of the objects and export the motion data at runtime to a local file. This file was then imported to *MATLAB*, where we retargeted the extracted sound grains according to these motion parameters and resynthesized the final audio offline.

All of the simulations were done with one Intel Core i7-3770 processor (3.40 GHz, 4 cores) using 8 threads. The simulation times for soft-body dynamics were typically below 30 seconds, except the jelly, which was our most challenging example and took around 30 minutes due to the large quantities of particles involved (over 200 per jelly). The computational times for the sound analysis and synthesis were typically between 4 to 5 seconds for impulsive sounds and 7 to 8 seconds for continuous sounds. Please watch the supplemental video #1[1] to view and hear the final results.

[1] https://youtu.be/Sn3wAVtatPg

3.3.1 Examples

Basketball Bouncing (Figure 8 (a)): We use *Bullet for Maya* as the physics engine for this simulation. A basketball (soft-body) is dropped onto a "hard" tile floor (with higher bounciness) and a "soft" carpet floor (with lower bounciness) from two different heights – 1 m and 2 m. Depending on the floor texture, this would generate different impact sounds.

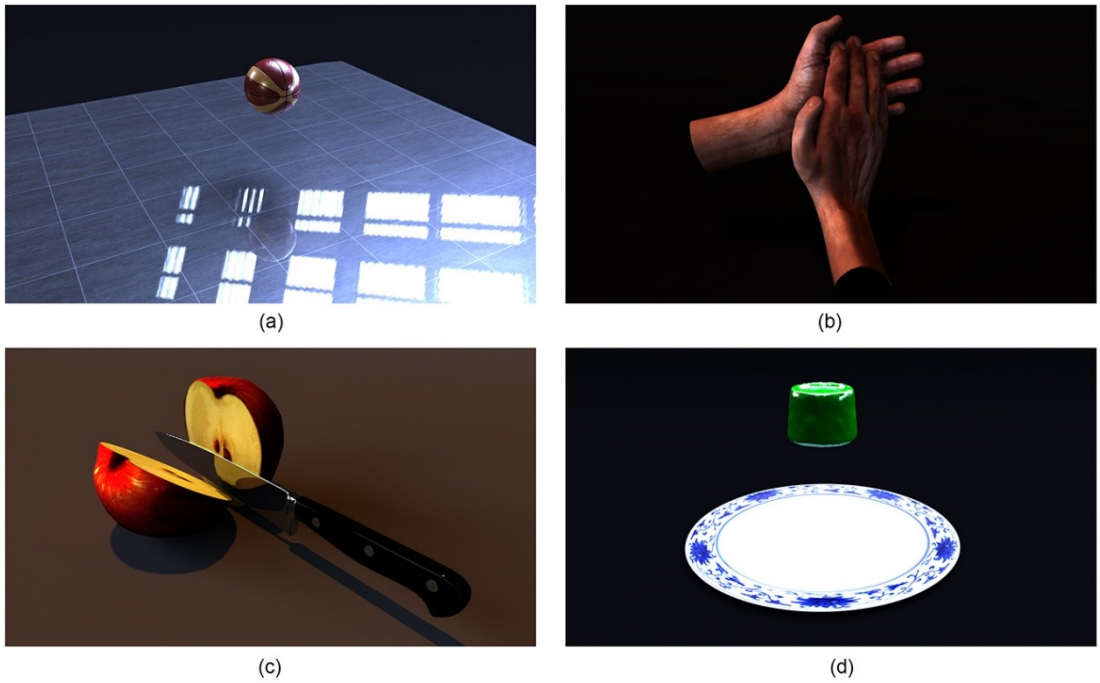


Figure 8 Examples of soft-body interactions. (a) Basketball bouncing, (b) hand clapping/rubbing, (c) apple slicing, (d) jelly dropping/shaking.

Hand Clapping/Rubbing (Figure 8 (b)): The hand clapping and rubbing actions are achieved through keyframe animation using the skeleton rigs attached to the hands. We animate all the clapping modes (see Figure 5) with different clap rates (1 s, 0.5 s, and 0.25 s intervals between each onset). To generate continuous sound, we create another

animation where two hands are rubbed against each other in parallel. There is a noted difference between resynthesized audio when the hands are rubbing at different speeds.

Apple Slicing (Figure 8 (c)): In this scene, an apple is cut in half with a kitchen knife. We animate the blade to be pushed straight down to a cutting board, slicing vertically through the apple at different speeds. There is a noted pitch shift in different examples. Due to the limitations of our pre-recorded database, we do not include the sound where the apple slice is dropped and hits the cutting board, or when the knife hits the board.

Jelly Dropping/Shaking (Figure 8 (d)): A jelly (created with *nParticles* from *Maya*) is dropped onto a ceramic plate at two different heights (15 cm and 30 cm). The plate is then animated to move from left to right for a few times to jiggle the jelly. This example demonstrates that our synthesis method is able to generate compelling audio for complex soft-body simulations with large elastic deformations. It also combines the generation for both impulsive and continuous sounds.

3.3.2 Preliminary Validation

In this section, we present a preliminary validation of the auditory perception of our examples by comparing the results obtained through our method with sounds produced from real-world recordings. A more comprehensive evaluation is provided in Chapter 5. To demonstrate that our synthesized sounds exhibit similar overall characteristics to their recorded counterparts, we look at their corresponding frequency spectra (impulsive sounds) and spectrograms (continuous sounds). Figure 9 shows the frequency

comparisons between the simulated outputs (black line) and the recorded data (red line) for impulsive sounds such as basketball bouncing and hand clapping. Due to the 100 Hz cutoff frequency in the high-pass filter we applied, the synthesized audio does not match the recorded one perfectly for low frequency range under 100 Hz. However, for the most perceptually significant frequencies (i.e., between 120 Hz and 8000 Hz), our results follow a similar evolution of the magnitude spectrum with respect to the frequency contents of the recordings.

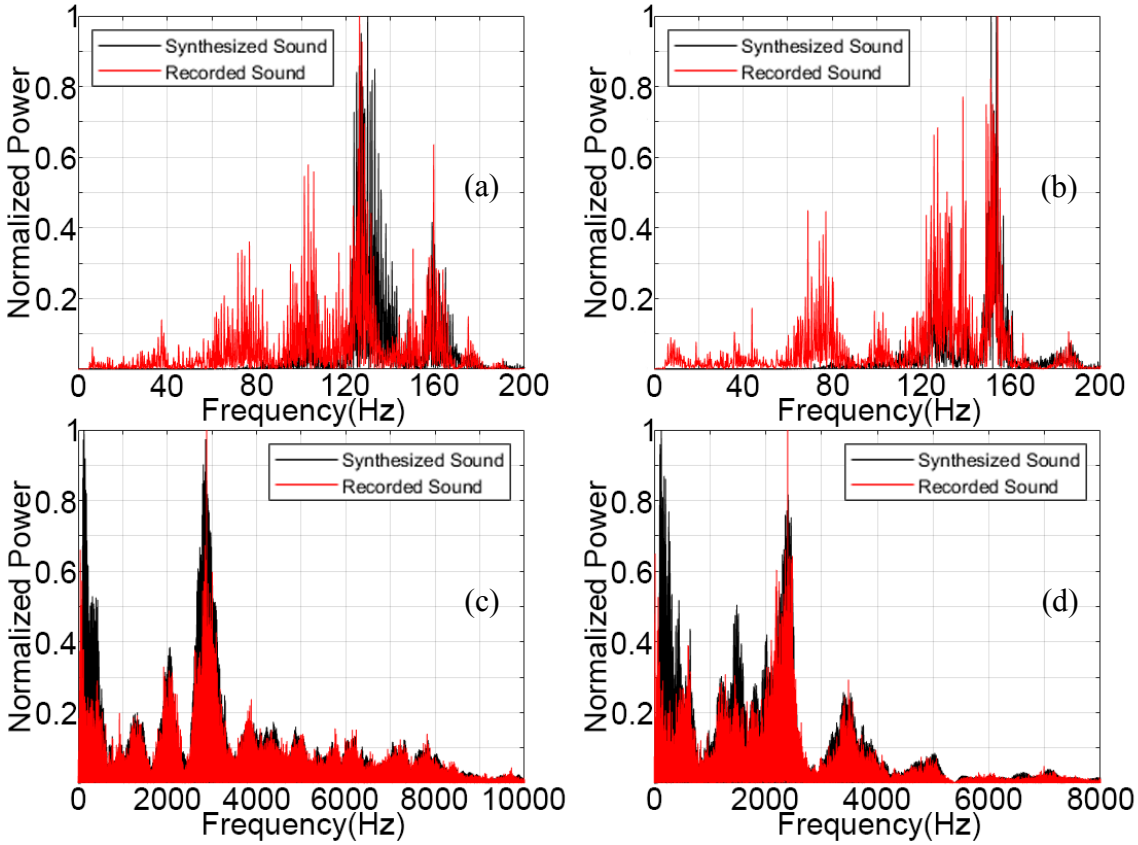


Figure 9 Frequency spectra for impulsive sound. (a) Basketball bouncing on hard floor, (b) basketball bouncing on soft floor, (c) hand clapping A2, (d) hand clapping P3.

For continuous sounds, as the resynthesized grains are the resampled versions of the original recordings, their frequency spectra would be changed. Therefore, we evaluated the quality of our generated audio by comparing the spectrograms between the two. Figure 10 shows the spectrograms of our simulated sounds (Left) and real recordings (Right) for apple slicing and jelly shaking as examples. Despite the clearly visible pitch/time shift, our synthesized sounds have a similar structural distribution of contact events as well as frequency ranges compared to real recordings.

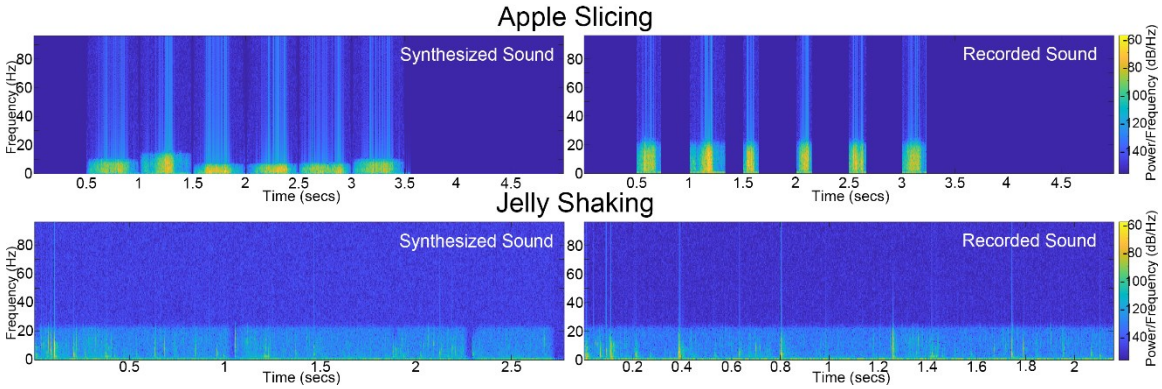


Figure 10 Spectrograms from synthesized and recorded versions of continuous sounds.

To further illustrate this, we provided both the synthesized and recorded sound examples in the supplementary video #1[2]. As shown in this video, these simulated sounds have little perceptually significant difference compared to the recordings. The reconstruction is quite good and appears to capture the significant sound textures well.

3.4 Summary

In this chapter, we have presented a data-driven approach for procedurally synthesizing plausible sounds for elastically deformable objects. Our method focuses on automatically

[2] https://youtu.be/Sn3wAVtatPg

extracting audio grains from real-world recordings, and then retargeting and resynthesizing these grains according to the motion data reported by the contact models. The grain extraction model we introduced is valid for typical elastic deformations that can produce impulsive or continuous sounds. Although our proposed method introduces some manual steps, and the synthesis process has to be performed offline due to the limitation of *Maya 2016* (which is not optimized for real-time audio processing), it still saves a great amount of effort over traditional fully-manual Foley artistry for animated movies. Our audio generation process is also computationally efficient and does not require large amounts of recordings. The above examples demonstrate its effectiveness for producing convincing soft-body sounds.

Chapter 4: Sound Synthesis for Plastic Deformation

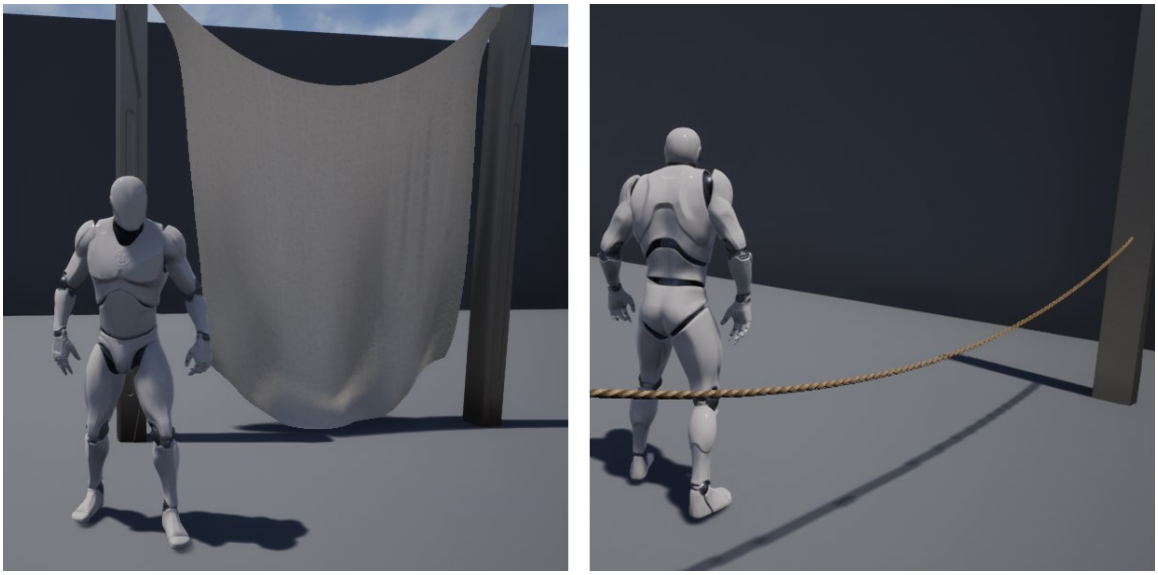


Figure 11 Common real-time soft-body dynamics with plastic deformations. Here we show a cloth hanging and a rope dangling.

4.1 Introduction

In this chapter, we investigate the sounds generated from the other type of soft-body interaction – plastic deformation. This kind of deformation is permanent, which happens after the deformation exceeds the elastic limit. Recent advances in computer graphics and games have made it possible to simulate realistic plastic deformations for many soft-body models (Figure 11). However, as mentioned in Chapter 2, synthesizing the corresponding audio for such deformations is still less explored. Unlike elastic deformations, plastic deformations can cause the velocity of the generated audio to change, thus simply modeling the elastic vibrations of the material is not sufficient to precisely capture the acoustic behaviors of these sounds. Additionally, due to the undergoing large and rapid

deformations, the produced sounds are often accompanied by complex aeroacoustic effects, such as rope swooshing. While there are some previous works that attempt to render such aerodynamic sound [46], [47], [55], no prior work exists for soft bodies.

To synthesize audio tracks for plastically deformable models, we propose an improved data-driven method based on the concept described in Chapter 3. Previously, our contact sound models only report the contact speed and energy of the entire object. To accurately capture the contact states of highly deformable (plastic) bodies, we develop more detailed parametric sound models which are able to perform shape and motion analysis at runtime and report contact information (such as speed and energy) at vertex level. Moreover, during the re-synthesis in the previous method, the continuous sound grains often have to be resampled to match the motion data from the deformations, which would create large pitch shifts that significantly alter the spectrum of the sounds if the undergoing deformations are too slow or too fast. It can also result in abrupt changes in pitch at the transitions between grains. As plastic deformations generally involve high-speed motions, we resolve this issue by using shorter sound segments in the database and concatenative synthesis since it offers more precise and higher level control. Finally, we combine the process for motion data extraction and grain retargeting to speed up the analysis and synthesize the sounds in real-time.

Our work also shares some similarities with An et al. [52], who synthesized cloth sounds with a motion-driven concatenative approach (see Section 2.2.3). However, their method requires a lengthy off-line pre-computation process and a degree of manual intervention,

which consumes more CPU power and is not suitable for real-time plastic deformations. In contrast, we use a different set of trade-offs and present a more efficient sound model that is capable of generating motion events at runtime and supports any particle-based plastic deformations. Furthermore, we also consider the associated aerodynamic sounds caused by rapid movement (such as rope skipping) which are missing from previous works. A more detailed comparison can be found in Section 4.4.3.

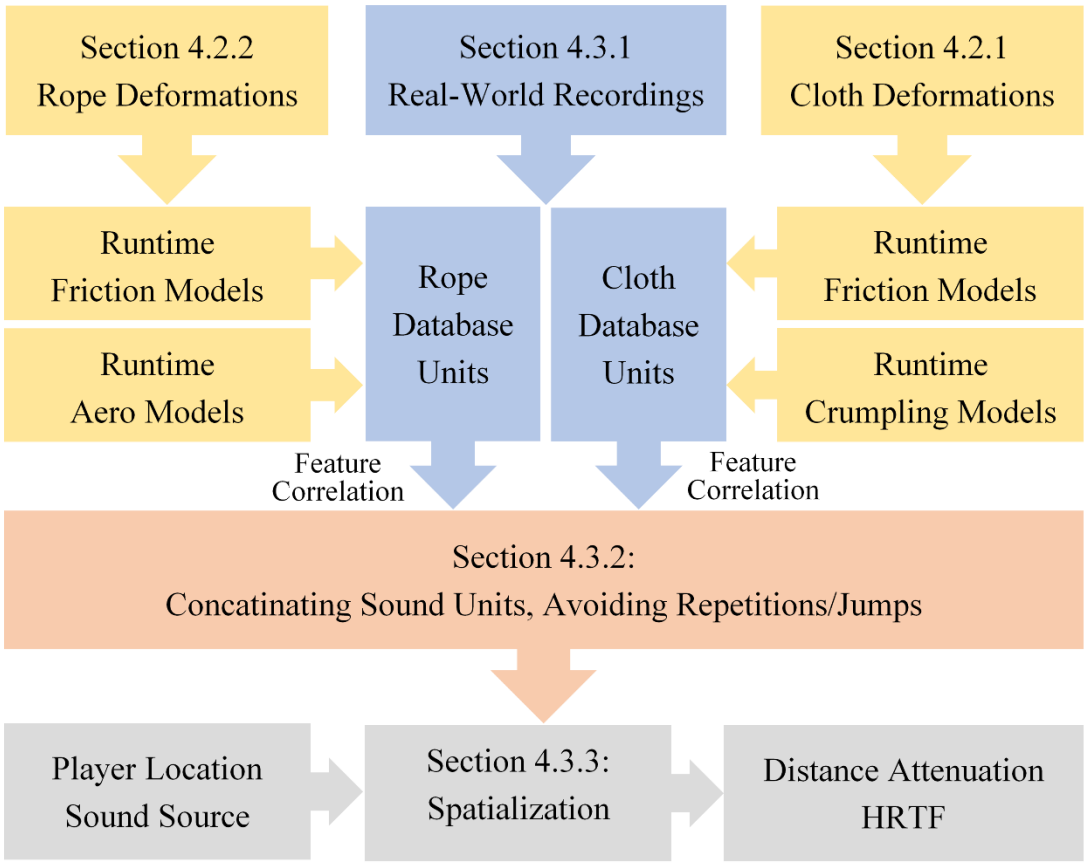


Figure 12 Overview of our data-driven synthesis approach for plastically deformable models.

Figure 12 shows an overview of our proposed system. First, we perform a geometric analysis of the object's plastic deformations at runtime, which can output several parametric sound models and detect sound producing events. Second, we pre-record a number of detailed sound databases for each object, and segment them into relevant short audio units. Next, based on the reported motion parameters from each sound model, we develop a real-time, automatic feature correlation technique to select the matching micro-sound samples from the corresponding database, and concatenate them into audio sequence. Finally, to further increase the realism of the results, we also add spatialization effects using a generic head-related transfer function (HRTF) and distance attenuation depending on the actual location of the player.

Our approach is capable of generating plausible plastic deformation sounds for soft bodies like cloth and rope in real time. Compared to existing CSS-based methods [52], [54], our system uses a more efficient and simplified parametric model to calculate the motions and contact states of plastic deformations, which significantly increases the computational speed and consumes less memory and CPU power. It also offers a variety of user-specified parameters that enable more control of the characteristics of the sounds. We demonstrate the effectiveness of our method on a cloth and rope simulation for *Unreal Engine 4* (*UE4*), with challenging interaction examples such as waving cloth in the wind and skipping rope.

4.2 Parametric Sound Models

In the previous chapter, we generated a range of sounds from the elastic deformations of four unique soft-body objects. Although these objects would also incur plastic deformations in certain scenarios, we normally do not consider that the associated sounds in most of the VE applications as the underlying elastic vibrations (caused by the plastic deformations) are very complicated and contain little or no audible frequencies. Therefore, to synthesize sounds from plastic deformations that are typical and familiar to us, we choose two new objects – cloth and rope, as our primary targets. These two types of soft bodies are available in most particle-based systems, and have been extensively researched in the computer graphics and gaming community [12], [91]. In this section, we describe the process of our synthesis method for both cloth and rope. We begin by detailing their parametric sound models.

4.2.1 Cloth Sound Model

Friction Sound. Cloth sounds usually consist of two components: a sliding friction sound and a buckling crumpling sound. Friction sounds are produced when cloth rubs against itself or other surfaces. Our friction sound models are similar to the one presented by An et al. [52], where we use sliding speed to approximate the pitch change of the friction sound. We also add the size of the contact region as another factor to affect the volume of the sound. These two parameters will be sufficient to model the sliding contact events of the simulated cloth.

Algorithm 1 Cloth Friction Model

1: **for** every particle $P_{i,t}$ on cloth **do**
2: get particle $P_{i,t}$ location $\boldsymbol{p}_{i,t}$
3: get player contacting location $\boldsymbol{q}_{i,t}$
4: **if** $\|\boldsymbol{p}_{i,t} - \boldsymbol{q}_{i,t}\| < \Upsilon$ **then**
5: **if** $P_{i,t} \notin C_t$ **then**
6: $C_t \leftarrow P_{i,t}$
7: **end if**
8: **end if**
9: **if** $\|\boldsymbol{p}_{i,t} - \boldsymbol{q}_{i,t}\| > \Upsilon$ **then**
10: **if** $P_{i,t} \in C_t$ **then**
11: $C_t \rightarrow P_{i,t}$
12: **end if**
13: **end if**
14: **end for**
15: **for** every contacting particle $P_{j,t} \in C_t$ **do**
16: get $P_{j,t}$ location $\boldsymbol{p}_{j,t}$
17: get $P_{j,t}$ location at previous frame $\boldsymbol{p}_{j,t-1}$
18: calculate sliding speed $\boldsymbol{v}_{j,t}$
19: **end for**
20: calculate average sliding speed $\boldsymbol{v}_{avg,t}$
21: get C_t size l
22: output $F(l, |\boldsymbol{v}_{avg,t}|)$

To estimate the sliding speed, we use a position-based collision analysis on each frame t of the simulation. For every particle (vertex i) on the cloth, we set up a finite collision radius Υ (Table 5). If any vertex on the player (mesh character) is within the distance Υ of the cloth particle, we add this particle $P_{i,t}$ to an array C_t as one of the contacting particles. If $P_{i,t}$ is no longer in contact with anything, we remove it from C_t. The result of the analysis is an array C_t that contains all contacting particles $P_{j,t}$. We then calculate the relative speed $\boldsymbol{v}_{j,t}$ of each contacting particle using a backward difference scheme. To avoid false contact detection, we discard any particle whose contacting speed is below a certain threshold s_t (Table 5). Finally, as an approximation of the size of the contact region, we also calculate the size (length) of the array C_t. As a result, we obtain a friction

sound model $F(l, |\boldsymbol{v}_{avg,t}|)$, where l is the array size $|C_t|$ and $\boldsymbol{v}_{avg,t}$ is the average speed of every contacting particle. Please refer to Algorithm 1 for more details.

Crumpling Sound. Crumpling sounds are generated when regions of the cloth surface suddenly change the bending direction. To model this kind of sound, we estimate the buckling events of a cloth by analysing its curvature changes. For each curvature change, we measure its energy to drive the synthesis of the crumpling audio.

To estimate the curvature of any particle on a cloth, we use a surface triangulation method [92], [93]. Given a particle $P_{i,t}$ at frame t, we consider all of its shared faces (triangles). Since each adjacent triangle is flat, the curvature is concentrated at the center of all triangles (their shared vertex), which is the particle itself. Suppose $P_{i,t}$ is shared by n edges, the scalar curvature K_j for each edge j can be estimated by

$$K_j = \frac{(\boldsymbol{n}_{i+1} - \boldsymbol{n}_{i-1})(\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1})}{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}\|^2} \tag{4.1}$$

where $\boldsymbol{p}_{i+1}$ and $\boldsymbol{p}_{i-1}$ are the positions of neighboring particle $P_{i+1,t}$ and $P_{i-1,t}$ (i.e. edge ends), and $\boldsymbol{n}_{i+1}$, $\boldsymbol{n}_{i-1}$ are the vertex normals respectively (Figure 13). Once all the edge curvatures are calculated, we take their average values as the mean curvature $H_{i,t}$ of particle $P_{i,t}$, namely

$$H_{i,t} = \frac{1}{n}\sum_{j=1}^{n} K_j \tag{4.2}$$

where n is the number of edges touching the particle. Finally, if the mean curvature of particle $P_{i,t}$ changes sign from frame $t-1$ to t, we measure its energy change using

$$E_{i,t} = (H_{i,t-1} + H_{i,t})^2 \tag{4.3}$$

After we process all the particles, we produce a list of buckling events C ($P_{i,t}$, $E_{i,t}$), with a total energy E to indicate the amplitude of the final crumpling audio. Please see Algorithm 2 for details.

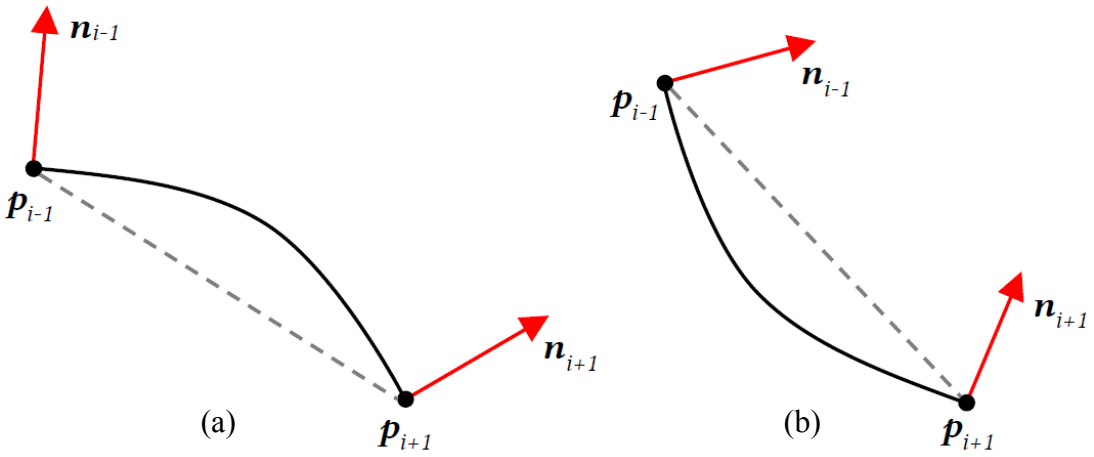


Figure 13 Calculation of edge curvature. If $K_j > 0$, we have a convex surface (a), and if $K_j < 0$, we have a concave surface (b).

Algorithm 2 Cloth Crumpling Model

$E_{total} \leftarrow 0$
for every particle $P_{i,t}$ on cloth **do**
    get all adjacent particle locations ($\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_{i+1}$)...
    calculate all vertex normals ($\boldsymbol{n}_{i-1}$, $\boldsymbol{n}_{i+1}$)…
    for each adjacent edge j **do**
    calculate K_j with (4.1)
    end for
    calculate $H_{i,t}$ with (4.2)
    if $H_{i,t} \times H_{i,t-1} < 0$ **then**
        calculate $E_{i,t}$ with (4.3)
    end if
    E_{total} += $E_{i,t}$
end for
output C ($P_{i,t}$, $E_{i,t}$), E_{total}

4.2.2 Rope Sound Model

Friction Sound. For rope, we usually identify three types of sounds: a sliding sound (friction), a stretching sound (squeak, twist), and a whoosh sound (swing, throw). The first two types of sounds can be modelled with external and internal friction sounds. For the last one, we use an aerodynamic sound model to parameterize such motion events.

Sliding contact sounds are produced when regions of the rope surface slide along another surface. Similar to the friction sound model of cloth (Algorithm 1), we estimate the average sliding speed $\boldsymbol{v}_{avg,t}$ of every contacting particles at frame t, and output an external friction model $F_{external}$ (l, $|\boldsymbol{v}_{avg,t}|$) to drive the synthesis of sliding sounds.

Algorithm 3 Rope Friction Model

```
→ Algorithm 1
get C_t size l
if l > 0 then
    Δs_t ← 0
    for every particle P_{i,t} on rope do
        get P_{i,t-1}, P_{i,t} locations p_{i-1,t}, p_{i,t}
        Δs_{i,t} ← ||p_{i,t} − p_{i-1,t}|| − s/n
        Δs_t += Δs_{i,t}
    end for
    calculate Δs_t' with (4.5)
end if
```

In addition to sliding, rope can also stretch and make creaking sounds due to the frictions between the twisted fibres inside. However, as most of the rope dynamics in commercial simulators are particle-based systems that do not simulate such internal frictions, we propose to measure the average rate of change in distance between each particle to model this kind of sounds (Algorithm 3). Given a rope with n particles and a rest length of s, we

look at how much the length of the rope changed after it has been stretched at each frame. If a player is in contact with the rope, this length change (absolute) can be simply calculated with

$$\Delta s_t = \sum_{i=1}^{n} \left(\left\| \boldsymbol{p}_{i,t} - \boldsymbol{p}_{i-1,t} \right\| - \frac{s}{n} \right) \tag{4.4}$$

where $\boldsymbol{p}_{i,t}$ is the location vector of particle $P_{i,t}$ at frame t. When the rope is not contacting anything, we set $\Delta s_t = 0$ so that it will not produce any sound when just dangling. Finally, we calculate the average distance change rate between each particle using

$$\Delta s_t' = \frac{\Delta s_t - \Delta s_{t-1}}{n \cdot \Delta t} \tag{4.5}$$

Here Δt is the time interval between frame *t-1* and *t*. This $\Delta s_t'$ will provide a rough approximation for the internal friction model, which is sufficient to synthesize the stretching sounds.

Aerodynamic Sound. When rope cuts through the air with a sufficiently fast speed, it periodically generates vortices (a whirling mass of air in a turbulent field) behind, creating subtle fluctuations of pressure that produce sounds (known as Aeolian tone). As a result, we can estimate the primary frequency of such aerodynamic sound using the frequency of the generated eddies.

For a circular cylinder moving through the air, the frequency of the generated Aeolian sound can be calculated with the following [46]:

$$f_{Aeolian} = \frac{\alpha v_{air}}{d_{cylinder}} \tag{4.6}$$

where v_{air} is the speed of the air near the cylinder surface, $d_{cylinder}$ is the diameter of the cylinder, and α is called the Strouhal number and is about 0.2 for the cylinder. As our rope object is one long cylinder divided by a series of particles, each segment can be treated as a small cylinder object that is connected to other segments via tensions and constrains. Therefore, each cylinder segment will be one sound source and we can use Equation (4.6) to approximate its frequency. However, as our current soft-body simulator does not simulate airflow around the rope, we have to use particle velocity to roughly estimate the air velocity. As a result, we obtain a set of *Aeo* ($P_{i,t}$, $f_{i,t}$) pairs for all rope segments at frame t, where $P_{i,t}$ is the particle at the end of each segment and $f_{i,t}$ is the predicted Aeolian tone frequency (see Algorithm 4).

Algorithm 4 Rope Aerodynamic Model

```
for every particle P_{i,t} on rope do
    get P_{i,t} location p_{i,t}
    calculate P_{i,t} velocity v_{i,t}
    if ||v_{i,t}|| > ϒ then
        calculate Aeolian frequency f_{i,t} with (4.6)
        Aeo_t ← f_{i,t}
    end if
    if ||v_{i,t}|| < ϒ then
        f_{i,t} ← 0
        Aeo_t ← f_{i,t}
    end if
end for
output Aeo (P_{i,t}, f_{i,t})
```

4.3 Concatenative Sound Synthesis

4.3.1 Pre-Recorded Sound Database

To synthesize the final audio using our parametric sound models, we constructed a detailed database of pre-recorded cloth and rope sound clips. As the characteristics of the sound produced depend on many factors such as the object's geometry shape, material and contact state, we need to record each sound under various different conditions. To simplify this process, we only selected a handful of different materials with different sizes as our database. For practicality, we also used a heuristic-based method to manually measure and acquire all of our experimental data (such as speed). All recordings were done in an anechoic chamber using an ultra-low-noise microphone (*RØDE NT2-A*) and a 192 KHz/24 Bit digital audio interface (same setup as described in Section 3.2.1).

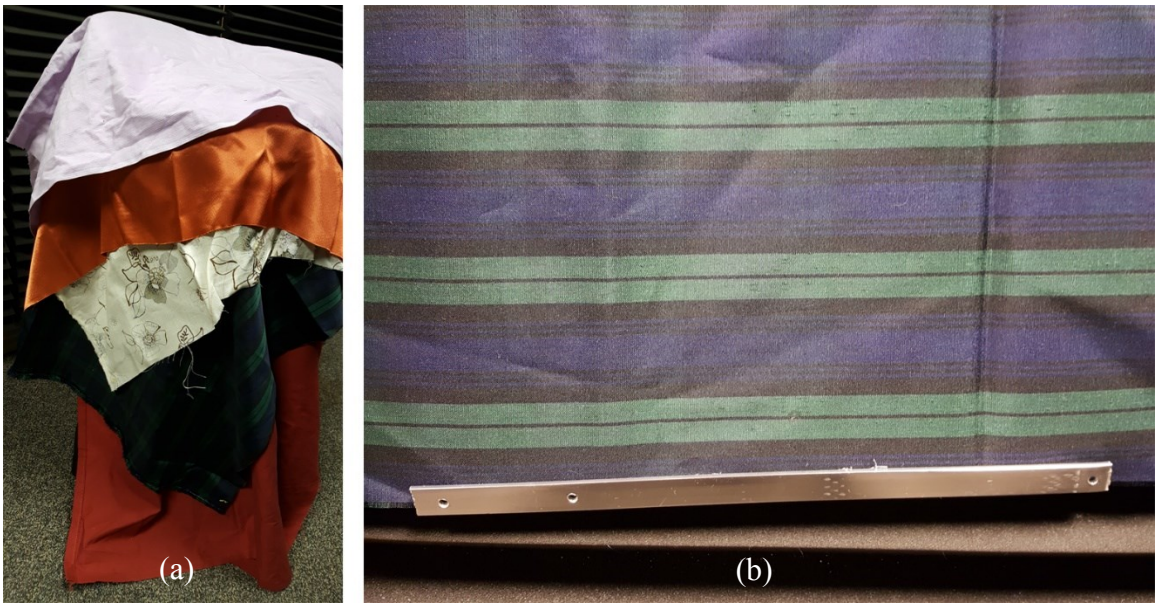


Figure 14 Cloth sounds recording setup.

Cloth Sound Database. For cloth sounds, we selected five types of common materials: corduroy, silk, linen, polyester, and cotton (Figure 14 (a), from top to bottom). To obtain recordings that can correspond to our parametric friction and crumpling models, we created two separate databases. For friction sounds, we hung a large square piece of cloth (100 × 100 cm) on the wall and attached a metal bar at the bottom to keep the surface flat (Figure 14 (b)). We then used three smaller pieces of cloth (1/10/25 × 25 cm, same material), and manually slid each piece against the large one at various speeds (approximately from 0.4 – 1.2 m/s). The resulting sounds were saved in a contact region size versus speed matrix, where each entry contained a 10-second continuous friction sound sequence.

To record crumpling sounds, we used a square piece of cloth in different sizes (10 × 10, 25 × 25 and 50 × 50 cm), and attached two metal bars at the edge of each cloth. We then manually sheared the cloth up and down with the metal bars at various frequencies (slow, medium and fast). This effectively created many distinct crumpling events with minimum frictions. The final crumpling sounds were stored in a matrix of cloth size versus energy (sum of squared amplitude values of recorded audio signals), where each entry typically had five to ten impulsive crumpling samples.

Rope Sound Database. We recorded rope sounds with two types of materials: sisal and polypropylene (PP) (Figure 15 (a), from top to bottom). Similar to cloth sounds, we constructed multiple databases specific to each parametric sound model. For external frictions, we hung one end of a rope (1.3 cm in diameter, 200 cm in length) from ceiling

and held the other end while using a shorter rope (50 cm in length, same material) to slide against it at various speeds (approximately 0.4 – 1.2 m/s). To produce stretching sounds, we used a pile of rope and stretched it back and forth from slow to fast. This would constantly create a series of internal frictions corresponding to the length change of the rope. Finally, we stored these data in their respective matrices for synthesis later.

Figure 15 Rope sounds recording setup.

To record aerodynamic sounds with respect to rope length and air speed, we swung the rope in circle while holding different spots on the rope (Figure 15 (b)). We began by segmenting a two-meter rope into four sections (each section is 50 cm). We then held the end of each section and slowly spun it in air with the wrist. While maintaining the grip, we progressively increased the speed until the rope was fast enough to make loud constant whoosh sounds. The whole process for each length was recorded as one long sequence (typically 10 to 15 s) and was segmented and stored in a matrix of rope length (50, 100 and 150 cm) versus spinning speed (roughly 0.8 to 4.0 cycles/s).

4.3.2 Unit Selection and Synthesis

Given the aforementioned parametric sound models and pre-recorded database, we are now able to synthesize the final sounds in real time for any cloth or rope object in a particle-based simulation system. Our method is adapted from CSS [53], where we use a unit selection algorithm to find and concatenate sample units from a source database so that they best match the target sound models. By generating macroscopic waveform structures from a large number of shorter ones, we can recreate meaningful sounds that preserve their original characteristics.

Database Segmentation. The first step of synthesizing sounds is to time-segment our source database recordings into units. As our databases are comprised of two types of sounds: continuous friction noises (cloth and rope) and discrete impulses (cloth crumpling and rope Aeolian tone), we need to treat these sounds in different ways. For continuous friction sounds, we extract a number of short clips (50 – 100 ms) from each entry of the database matrix, and convolve them with a triangular window. These segments are then stored back as a sub-matrix in the original one, where we also attach a label for each clip (Table 3). For impulsive sounds, we simply extract each discrete impulse according to certain input amplitude threshold α (described in Section 3.2.3). If any signal amplitude exceeds α (we set $\alpha = 0.05$), we extend each signal to its nearest zero-crossings, and convolve it with a rectangular window. We then label and store all the decomposed signals back as sub-matrices. These symbolic data can later be exploited for unit selection.

Table 3 Sound model parameters and database labels.

Soft Bodies	Sound Models	Database Labels
Cloth	$F\ (l,\ \lvert \boldsymbol{v}_{avg,t} \rvert)$	Contact Region Size s, Sliding Speed v
	$C\ (P_{i,t},\ E_{i,t}),\ E_{total}$	Cloth Size s, Crumpling Energy E
Rope	$F_{external}\ (\lvert \boldsymbol{v}_{avg,t} \rvert)$	Sliding Speed v
	$F_{internal}\ (\Delta s_t')$	Stretching Speed v'
	$Aeo\ (P_{i,t},\ f_{i,t})$	Rope Length l', Spinning Speed ω

Unit Selection. The next step is to select the database units for each target sound model, and concatenate them together to form the final audio. An overview of the entire synthesis process is given in Algorithm 5, which we will explain in more detail below.

Algorithm 5 Overall Sound Synthesis Process

get $(l,\ \lvert \boldsymbol{v}_{avg,t} \rvert)$ or $\Delta s_t'$ (Algorithm 1, 3)
select unit $(s,\ v)$ or v' with (4.7) – (4.9)
skip for 1/2 unit duration
if large jumps detected **then**
    select new unit
end if
concatenate selected units with 50% overlap
get $(P_{i,t},\ E_{i,t})$ or $(P_{i,t},\ f_{i,t})$ (Algorithm 2, 4)
if $(\lvert E_{total,t} - E_{total,t-1} \rvert > 10)\ \|\ (E_{total,t} / E_{total,t-1} > 2)$ **then**
    select unit $(s,\ E)$ with (4.7) – (4.9)
end if
if $(f_{max,t} < f_{max,t-1})\ \&\ (f_{max,t} < f_{max,t+1})$ **then**
    select unit $(l',\ \omega)$ with (4.7) – (4.9)
end if

As shown in Table 3, given any output sound model, we need to find the corresponding database units according to their labels. For speed/energy related datasets, we can approximate the correspondence between source and target using linear interpolations.

Suppose we have an output model parameter x_t (x_t can be any of $|\boldsymbol{v}_{avg,t}|$, $E_{i,t}$, Δs_t', or $f_{i,t}$), where it has a range of of $x_{t,min}$ and $x_{t,max}$ (Table 5). For any database unit j labelled with u_j (u_j can be v, E, v', or ω, respectively), whose range is $u_{j,min}$ and $u_{j,max}$, we assume there exists a one-to-one mapping:

$$u_j' = J(x_t) = u_{j,min} + (x_t - x_{t,min})\frac{u_{j,max} - u_{j,min}}{x_{t,max} - x_{t,min}} \tag{4.7}$$

Next we simply use the nearest neighbour of u_j' to select the matching unit of x_t with the label u_t, where

$$u_t = \arg min_j \left\| u_j' - u_j \right\|^2 \tag{4.8}$$

where j is the index of each database (matrix) entry.

While linear interpolation works for datasets that are evenly distributed, we also have other data where the source and target distribution differs significantly. For example, all the size/length related database units have very limited entries, which are not enough to map all the possible outputs from our sound models. Therefore, we introduce a user-defined correspondence to manually warp the source units into the space of the target model parameters. Given an output parameter y_t (y_t can be l or $\sum P_{i,t}$) with a range of $y_{t,min}$ and $y_{t,max}$, we specify a few intervals (Δ_1, Δ_2, …, Δ_n), where n is the total number of related database labels u_k (s or l') and $\Delta_1 + \Delta_2 + \ldots + \Delta_n = y_{t,max} - y_{t,min}$. This enables us to setup a multiple-to-one mapping:

$$\left(y_{t,i} + \Delta_k\right) \rightarrow u_k \tag{4.9}$$

so that if y_t falls under any of the interval, we can always find a database unit with the label u_k to be our matching sound. The length of each interval Δ varies depending on how the user specifies them.

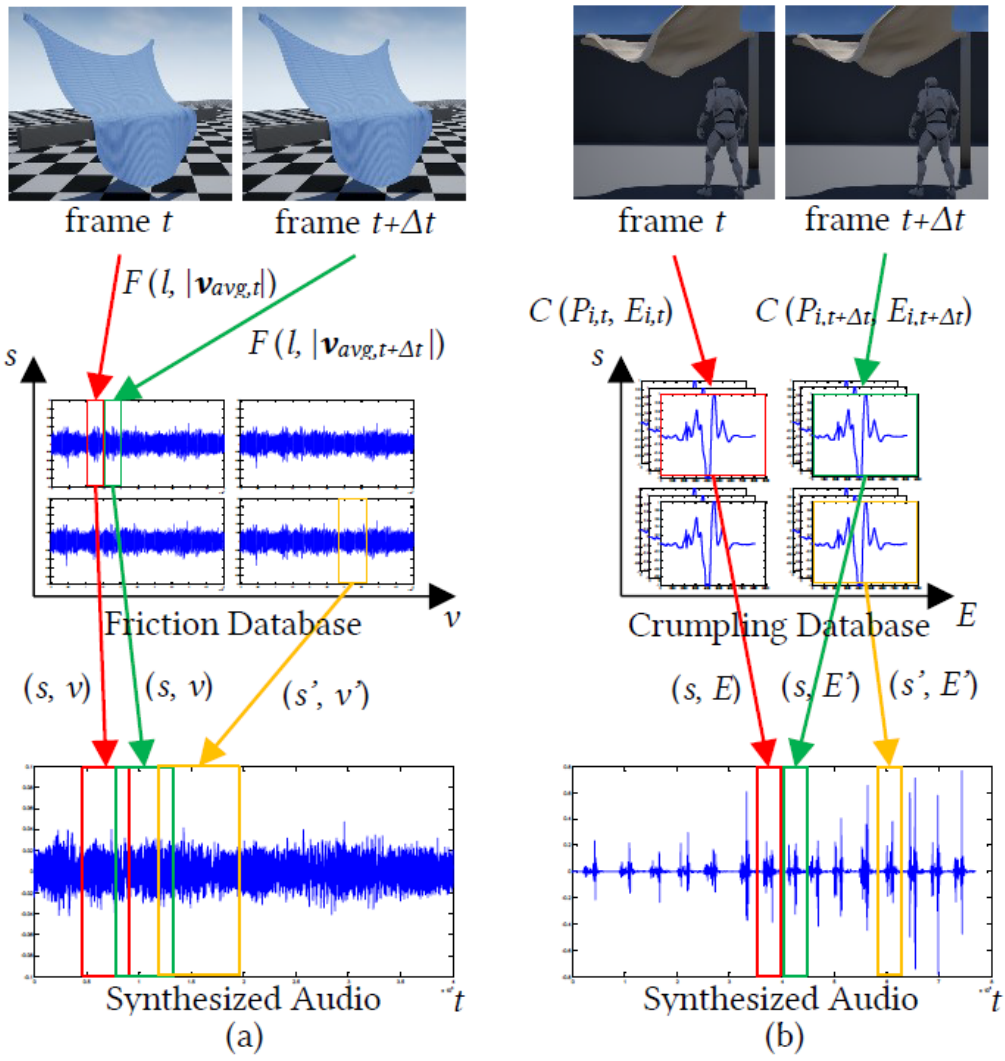


Figure 16 Sound synthesis. (a) Synthesizing cloth friction sounds, (b) synthesizing cloth crumpling sounds.

Sound Synthesis. To synthesize friction sounds for cloth (Figure 16 (a)), we constantly monitor the output of our friction model $F(l, |\boldsymbol{v}_{avg,t}|)$ at each gameplay frame t. For parameter l, we typically specify three intervals $\Delta_1 = 100$, $\Delta_2 = 200$, and $\Delta_3 = 400$, which correspond to the three different cloth sizes we have in our database (Section 4.3.1). Given any $(l, |\boldsymbol{v}_{avg,t}|)$, we use Equation (4.7), (4.8) and (4.9) to select the corresponding

unit with the label (s, v). We then skip over for a few frames (which equal to 1/2 of the selected clip length) and begin to overlap and add the next one. If, during the skipped time, the output data at two consecutive frames have large jumps (i.e. $|\boldsymbol{v}_{avg,t}| / |\boldsymbol{v}_{avg,t-1}| > 10$), we immediately fade out the current clip and switch to the new matching unit for $|\boldsymbol{v}_{avg,t}|$. The final output signal is a sequence of many short units with 50% overlap. To avoid repetition and large discontinuities between units, all the database units under the same label are stored as continuous sequences. If multiple units are selected with the same label, they tend to be synthesized as consecutive units from the database. Additionally we setup a random starting unit so that same sequence will not be repeated. This helps to preserve more temporal structure of the original recordings.

For crumpling sounds of cloth (Figure 16 (b)), we look at the total curvature energy change at each frame. Given a list of extracted buckling parameters C ($P_{i,t}$, $E_{i,t}$), we first calculate the total number of "crumpling" particles $P_{i,t}$, which is used to find the unit with the matching cloth size (here the specified intervals are $\Delta_1 = 10$, $\Delta_2 = 50$, and $\Delta_3 = 100$). Next, we compare the total curvature energy E_{total} at two consecutive frames t-1 and t to look for the crumpling event. If this energy change exceeds certain threshold (we find that either $| E_{total,t} - E_{total,t-1} | > 10$ or $E_{total,t} / E_{total,t-1} > 2$ works best), we assign a recorded database unit that matches this $E_{total,t}$ with Equation (4.7) and (4.8). To avoid repetition, we store multiple (5 to 8) units under the same label. Finally, we discard any sound with $E_{total} < 1$, as we assume that the cloth with such total curvature energy would be in an idle position and thus would not produce any audible sound.

Similarly, we synthesize the sounds for rope. For external and internal frictions, we apply the same method described above except that we do not utilize any information regarding to the size of contact region, as we find in our experiment that different contact region sizes do not produce sounds that are perceptually different. For aerodynamic sounds, we output *Aeo* ($P_{i,t}$, $f_{i,t}$) for each rope segment i at frame t. The total number of $P_{i,t}$ serves as an indication of the rope length. We then look for the maximum Aeolian frequency $f_{max,t}$ of all segments, and set the corresponding segment as our primary sound source. If, during any three consecutive frames, we have $f_{max,t} < f_{max,t-1}$ as well as $f_{max,t} < f_{max,t+1}$ (indicating the start of an Aeolian tone at t), this frequency will be used to select the appropriate sound clip from our database. We find that this approach is effective for any fixed rope that can exhibit strong vibrational response.

4.3.3 Spatialization

To provide the player with a sense of space in the 3D world, we add some spatialization effects to the synthesized sounds by taking into account the distance attenuation and listener's actual position. This offers more information to the player about the direction and source of the sounding object, which adds another layer of realism to the generated audio.

After synthesis, the sound will be played at the location where the motion event occurs. For friction sounds, this would be the position of the contacting particle. For crumpling and aerodynamic sounds, this would be the position of the particle with $E_{max,t}$ or $f_{max,t}$. Effectively, we have many moving point sources that constantly broadcast audio signals.

To attenuate the sound in regards to the distance *d* between player and sound source, we setup two border radii R_{min} and R_{max}, and use the following distance function:

$$A(d) = \begin{cases} A_0, & d \leq R_{min} \\ A_0 10^{-d/20}, & R_{min} < d < R_{max} \\ 0, & d \geq R_{max} \end{cases} \quad (4.10)$$

where A_0 is the original amplitude and $A(d)$ is the attenuated one. For all of our examples, R_{min} is typically 3 – 4.5 m, and R_{max} is typically 18 – 20 m. This attenuation model offers a more natural falloff model that attempts to simulate how we actually hear sounds in a normal environment.

Furthermore, for binaural auralization, we implement an HRTF system using the model described by Brown and Duda [94]. Based on the position of the sound source, the system filters and attenuates the sounds and applies these changes to the left and right channel of the stereo audio while taking into account the head model of the listener. This setup helps to create a convincing sense of location for the sounds generated.

4.4 Results and Discussions

4.4.1 Implementation Details

Our system was implemented in *Unreal Engine 4* (version 4.19.2). For our soft-body simulator, we used a commercially available *UE4* plugin called *VICO Dynamics*, which was a particle-based simulation system running on the CPU. The parametric sound models were integrated in the *Unreal Blueprint Class* for each cloth and rope object using both visual scripting and C++. To increase runtime efficiency, each sound model will output its motion data in sequence for every event tick. All of our pre-recorded database

units were imported as 16-bit stereo wave files with a sample rate of 44.1 KHz (as recommended in the *UE4* documentation), and stored in the *Sound Cue*, where the spatialization effects were added with the *Steam Audio* plugin. The unit selection and synthesis process was also done in the object's *Blueprint Class*, which constantly concatenated the corresponding sound units according to the motion data, and played the final audio at the event location.

Table 4 Performance results. Here F denotes friction sound, C denotes crumpling sound and A denotes aerodynamic sound.

Cloth/Rope Type	Database Size	Max Audio Memory Used	Avg. Synthesis Time
Cotton	2.45 MB (F) 1.77 MB (C)	2.85 MB	5.64 ms (Fig. 17 (a)) 6.45 ms (Fig. 17 (c))
Linen	2.60 MB (F) 1.99 MB (C)	4.16 MB	5.62 ms (Fig. 17 (a)) 6.34 ms (Fig. 17 (c))
Polyester	2.67 MB (F) 2.11 MB (C)	3.51 MB	5.85 ms (Fig. 17 (a)) 6.25 ms (Fig. 17 (c))
Silk	2.80 MB (F) 2.24 MB (C)	4.61 MB	5.60 ms (Fig. 17 (a)) 6.21 ms (Fig. 17 (c))
Corduroy	2.67 MB (F) 2.04 MB (C)	4.28 MB	5.57 ms (Fig. 17 (a)) 6.36 ms (Fig. 17 (c))
PP	1.97 MB (F) 1.31 MB (A)	3.24 MB	0.57 ms (Fig. 17 (d)) 0.75 ms (Fig. 17 (f))
Sisal	2.40 MB (F) 1.88 MB (A)	3.68 MB	0.50 ms (Fig. 17 (d)) 0.78 ms (Fig. 17 (f))

Table 4 shows the database size for each object. To test the stability of our system, we ran our demos on a number of different hardware (from low-end to high-end), all of

which were able to maintain a smooth frame rate between 60 to 120 FPS, with a relatively low CPU load for both simulation and sound synthesis (typically below 15% for cloth, and 8% for rope). Table 4 gives the performance results on a mid-end hardware (3.40 GHz, 4-core *Intel Core i7-3770*, 16 GB DDR3 RAM, *NVIDIA GeForce GTX 1060* with 6GB VRAM). As our parametric sound models were frame rate dependent, we found that higher frame rate would typically yield to better results.

Table 5 Example parameters. Here Υ is collision radius, s_t is valid speed threshold, (x_{min}, x_{max}) is the range of sound model.

Cloth/Rope Type	**Υ (cm)** **s_t (cm/s)**	**(x_{min}, x_{max})**
Cotton	0.5 1	$(1, 800)_{\|v\|}$ $(0.001, 100)_E$
Linen	0.5 1	$(1, 800)_{\|v\|}$ $(0.001, 100)_E$
Polyester	0.5 0.5	$(0.5, 1000)_{\|v\|}$ $(0.0001, 200)_E$
Silk	0.5 2	$(2, 800)_{\|v\|}$ $(0.001, 80)_E$
Corduroy	0.5 1	$(1, 800)_{\|v\|}$ $(0.001, 150)_E$
PP	0.1 0.5	$(0.5, 1000)_{\|v\|}$ $(15, 500)_{\Delta s'}$ $(10, 300)_f$
Sisal	0.1 0.25	$(0.25, 1000)_{\|v\|}$ $(15, 500)_{\Delta s'}$ $(20, 200)_f$

4.4.2 Examples

We now present a variety of examples that demonstrate the effectiveness of our methods. All of the characters, models, materials, animations and lightings that we used were from the *UE4* starter content (*Third Person Template*). The textures (including normal maps) for the cloth and rope were from *TurboSquid* (https://www.turbosquid.com/) and *3D TEXTURES* (https://3dtextures.me/, with CC0 license). Specific simulation and synthesis parameters for each object are given in Table 5. Please view our supplemental video #2[3] with stereo headphone for a more detailed result.

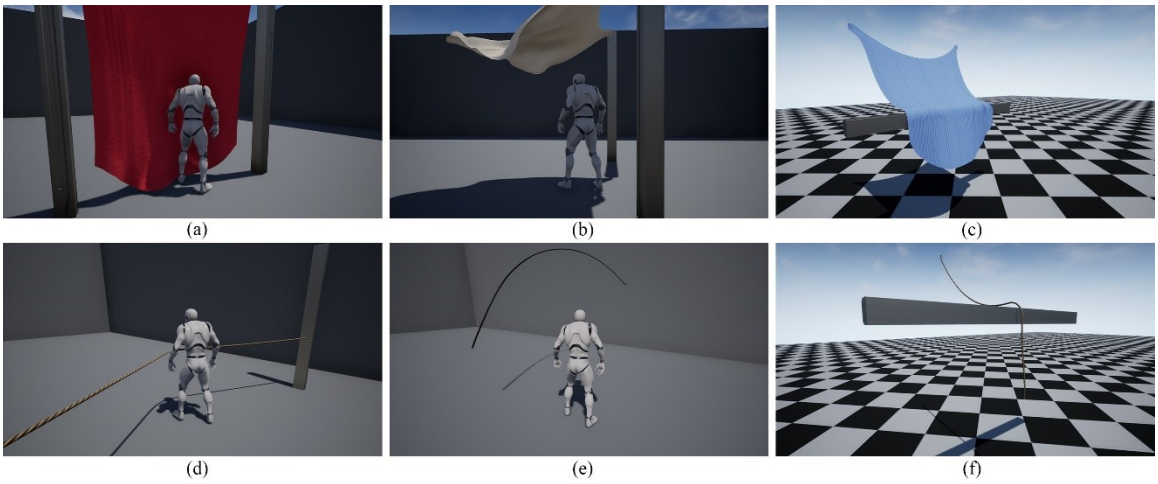


Figure 17 Examples. (a) Cotton sheet with character, (b) linen sheet waving, (c) corduroy sheet sliding, (d) rope dragging, (e) rope skipping, (f) rope sliding.

Cloth with Character (Figure 17 (a)). A large piece of cloth (3 × 3 m, with 30 × 30 particles) is hung between two pillars. Player can control a robot-like character to interact with the cloth freely. The material of the cloth can be switched by using a pause menu to place the character in different levels. To optimize the performance, we limit one type of cloth per level. The resulting sounds from each type of material are noticeably different.

[3] https://youtu.be/p87DudRTBvE

Furthermore, we setup a directional wind source that can be turned on and off. When the player activates the wind, the affected cloth would exhibit a series of high speed buckling events, thus producing fast crumpling sounds similar to a waving flag (Figure 17 (b)). Additionally, we add a 90-second looping background noise and footstep sounds when the player's character is moving, which helps to enhance the immersion in the 3D environment.

Cloth with Pole (Figure 17 (c)). We use the same pieces of cloth as previous example and hang each piece at two fixed points in the air for each level. This time, the player can control a pole (0.2 × 0.2 × 4 m) to slide against the cloth with different speeds (player can use keyboard to change the moving speed of the pole from 0.6 to 7 m/s). This allows us to create more low-speed continuous sliding motions on the cloth. As a comparison, we also record a video using the same setup with real cloth (please refer to the supplementary video #2[4]). Additionally, we have an extra level where we can drop a series of different types of cloth on the ground and create loud crumpling sounds.

Rope with Character (Figure 17 (d)). Similar to cloth, we hang a rope (diameter: 0.025 m, rest length: 5 m, number of segments: 25) between two pillars, and the player can control a character to interact with the rope. We can notice the characteristic stretching sound when the character is dragging the rope with its body and the pitch-changing aerodynamic sounds when the rope is released and vibrate back and forth. Moreover, we create a skipping rope in another level with one end of the rope fixed and the other end attached to a short rotating rod (Figure 17 (e), where the rod is hidden in the scene). By

[4] https://youtu.be/p87DudRTBvE

adjusting the rotational speed of the rod, we can change how fast the skipping rope moves in air, thus creating different Aeolian tones. We also compare this sound with a real skipping rope.

Rope with Pole (Figure 17 (f)). In this scene, we use a shorter rope (4 m) with more segments (50). A moveable pole can be controlled with various speeds (0.6 – 7 m/s) to slide against the rope. Player can select different types of rope by switching to different levels. Similar to cloth, this enables us to generate many slow sliding events which would produce noticeable friction sounds. We can also note the subtle spatialization effects in the final results when the rope is moving towards or away from the camera.

4.4.3 Discussions

Comparison to Real Recordings. To preliminarily validate the perceptual quality of our examples, we compared our synthesized results directly to real-world recordings. Again, for a more comprehensive evaluation, please see Chapter 5. As references, we selected four scenes (cloth sliding with pole, cloth dropping, rope sliding with pole, and rope skipping) from our examples, and recorded them with real cloth and rope in a sound isolation room. To acquire the best sound quality, all audios were recorded separately (using the same setup as we construct our database) and synchronized with the videos afterwards.

As shown in the supplemental video #2[5], overall our synthesized sounds exhibit similar characteristics compared to the recorded ones. Specifically, for frictional contacts, we

[5] https://youtu.be/p87DudRTBvE

have more high-pitched loud sounds from the polyester and corduroy material, especially when the cloth quickly slips off the pole, making a pitch-changing “zipping” effect. Additionally, the sliding sound from sisal rope would generate many subtle “sizzling” noises due to its highly uneven textures. In contrast, for material with smooth surface (such as silk or PP rope), the sounds tend to be more low-pitched and quiet.

For the cloth dropping test, we observed that the recorded and synthesized sounds of each type of cloth would produce different characteristics. Our simulated sounds have more low-frequency content, whereas the real sounds contain more impulsive features, and are much richer and varied. This is partially due to fact that our cloth simulator does not consider the air resistance associated with the falling of the cloth, which results in completely different contact motions compared with the real cloth. Another factor is that the recorded dropping sounds are actually made of multiple kinds of sounds including some crumpling noises, a loud impact sound from the ground, and some small internal friction noises, while our synthesized audio only involves some large crumpling noises. Nevertheless, the simulated dropping sounds are still plausible and synchronized well with the motion of the cloth.

Last but not least, for aerodynamic sounds, we can clearly hear that when we speed up the rope skipping, the Aeolian (swoosh) sound quickly shifts up in pitch due to the increase of the frequency, just like what we have simulated in the game. It should also be notified that although the subtle spatialization effects are present in the simulated audio, we do not take into account the Doppler effects that may also change the frequency of the

sound in relation to the moving listener. Therefore, the results are somewhat different compared to the recorded ones, especially when the sound source is moving away from the recording device/camera.

Comparison to Previous Works. As mentioned in Chapter 2, some previous work on cloth sound synthesis exist. Here we highlight the similarities and differences compared to these works. O'Brien et al. [22] proposed to use FEM for physically-based sound synthesis, where cloth examples were also considered. However, the main drawback for this method is the high computation time (over 20 hours) and complex parameter tuning (over 1,000 elements). Therefore, we take a different approach by using a data-driven method, which is easier and more practical for real-time synthesis.

Our work is more similar to An et al. [52], where both works use a friction/crumpling sound model and a manual feature warping technique to drive the synthesis of cloth sounds from real recordings. Beyond this, however, the focuses of these two works are different. The synthesis process of An et al. consists of two steps: First, they generate a low-quality target signal from the parametric sound models. Second, they use CSS to piece together cloth recordings to best match the target signal. However, their mapping process relies on a sound designer who provides a number of manual correspondences between the target and database signal. Although high fidelity sounds can be generated for cloth animations, their approach has slow simulation time (typically 5 to 30 minutes), lengthy manual correspondence (it takes around 5 to 15 minutes, with 2 – 5 iterations), and is limited to only a small number of given animations.

On the other hand, our method focuses on real-time sound synthesis for common plastically deformable bodies. We consider a more generalized approach that we believe can also be applied to other particle-based soft-body objects other than cloth and rope. Instead of generating low-quality target sounds at first, we propose a number of simplified sound models that are used to directly drive the synthesis of recorded databases, which significantly speeds up the computation time with some trade-off in the final sound qualities. Although our manual feature correspondence cannot achieve the same level of fidelity as An et al., it only requires the user to specify a few parameters (such as time intervals or energy thresholds) within 3 – 5 seconds, is performed once, and can be reused for the same type of object. We also demonstrate that our approach is able to handle a variety of interaction scenarios, including some challenging ones such as cloth waving and rope skipping. In general, all three works address distinct aspects of sound synthesis for cloth (physics model vs. data-driven animation vs. real-time simulation/game, respectively) and have different application fields.

4.5 Summary

In this chapter, we have presented a real-time system for synthesizing plausible sounds for plastic deformations in soft bodies. To analyze the geometric deformations of the object, we develop a number of simplified parametric sound models that are able to extract specific sound-producing motions at any frame. These data are then used to select and synthesize micro sound units from a pre-recorded audio database. By implementing this system with cloth and rope in *UE4*, we have demonstrated that our method is valid

for many typical plastically deformable interactions, and ideally could be extended to work with any particle-based systems. It also consumes less memory, and is more computationally efficient compared to prior data-driven synthesis methods for soft bodies.

Chapter 5: Evaluation of Synthesized Sound

5.1 Introduction

In previous chapters, we have preliminarily evaluated the qualities of our synthesized results by directly comparing them with real recordings via videos/frequency spectra/spectrograms (see Section 3.3.2, 4.4.3). Such methods are common in rigid-body sound evaluations [95], and can provide some information regarding to the structural distribution and frequency ranges of the generated soft-body sounds. However, our hearing system is both objective and subjective. To comprehensively evaluate the synthesized sounds, we also need to incorporate the understanding of human auditory perception. Moreover, all of the aforementioned validation techniques do not measure the similarities between the synthesized and recorded audio. Therefore, they cannot be used as standard metrics to effectively and accurately measure the objective quality of our results.

Another issue with evaluating the synthesized sounds of soft bodies is that typically the audio generation process creates many subtle variations from the original recorded database. The output sounds have similar frequency content but their temporary structures may be completely different, which would generally yield poor results using traditional audio fingerprinting method (see Section 2.4). Thus, we require more advanced evaluation techniques that are able to identify these variations and efficiently capture the characteristics between these sounds. As discussed in Chapter 2, the recent advance of deep learning technology in audio signal processing has dramatically

improved its performance in extracting the underlying features of audio, which makes deep learning well-suited for evaluating complex soft-body sounds.

In this chapter, we investigate both subjective and objective evaluation criteria to comprehensively assess the performance of our procedurally-generated audio, including both elastic and plastic deformation sounds from Chapters 3 and 4. We also seek to compare our synthesized cloth sounds to An et al.'s results [52], as both synthesis methods are closely related to each other (discussed in Section 4.4.3). For subjective evaluation, we conduct a three-part perceptual study to explore: (a) the recognisability of our synthesized sounds, (b) the quality of synthesized sounds compared to real-world recordings, and (c) the synchronization of sound and motion. In our objective evaluation, we borrow the concept of Inception Score (IS) [69] and Fréchet Inception Distance (FID) [87] from the assessment of generative adversarial networks (GANs) performance, and adapt them to measure the recognisability and similarity of our synthesized sounds to real ones.

5.2 Subjective Evaluation

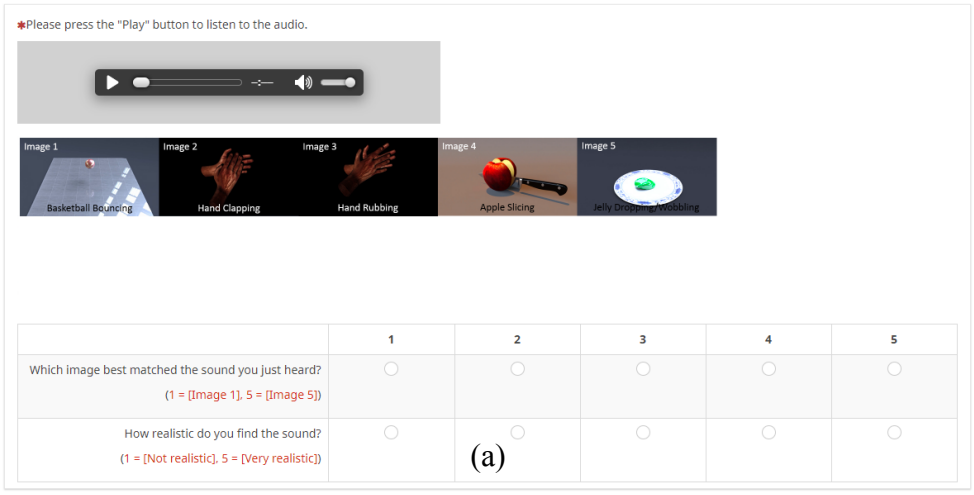
To evaluate the subjective expressiveness of procedurally-generated audio for soft-body interactions, we conducted perceptual studies adopted from previous evaluation techniques [25], [35], [45], [60], [61], where recognisability, quality, and synchronization of the synthesized sounds were assessed.

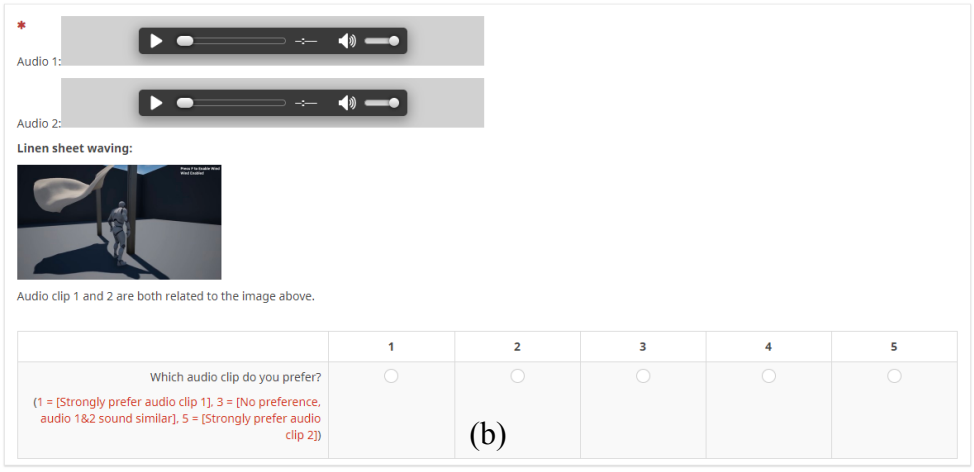
5.2.1 Participants

A total of 25 participants (12 females and 13 males) participated in our studies with informed consent, which was approved by the Carleton University Research Ethics Board B. Their ages ranged from 21 to 43 (mean = 29.84, SD = 4.26). All of the participants reported no significant visual or auditory impairment. Participants were also asked to rate their familiarities with VEs on a scale of 1 to 5, with 5 being very familiar and 1 being no experience. The responded mean score was 3.12, with a SD of 1.24.

Table 6 List of selected sound categories/labels with all the stimuli, grouped by synthesis method.

Group 1	**ball_bouncing**	**clapping**	**hand_rubbing**	**apple_slicing**	**jelly**
Stimuli	high_hardfloor	clapping_fast	rubbing_fast	slicing_fast	drop_high
	low_hardfloor	clapping_med	rubbing_med	slicing_med	drop_low
	high_softfloor	clapping_slow	rubbing_slow	slicing_slow	wobbling
	low_softfloor				
Group 2	**cloth_waving**	**cloth_sliding**	**rope_spinning**	**rope_sliding**	
Stimuli	cotton_waving	cotton_sliding	sisal_spinning	sisal_sliding	
	ploy_waving	poly_sliding	pp_spinning	pp_sliding	
	linen_waving	linen_sliding			
	silk_waving	silk_sliding			
Group 3	**cotton_An**	**poly_An**	**boxing_An**		
Stimuli	cotton_sheet	poly_sheet	boxing_fast		
	cotton_couch	poly_couch	boxing_med		
			boxing_slow		





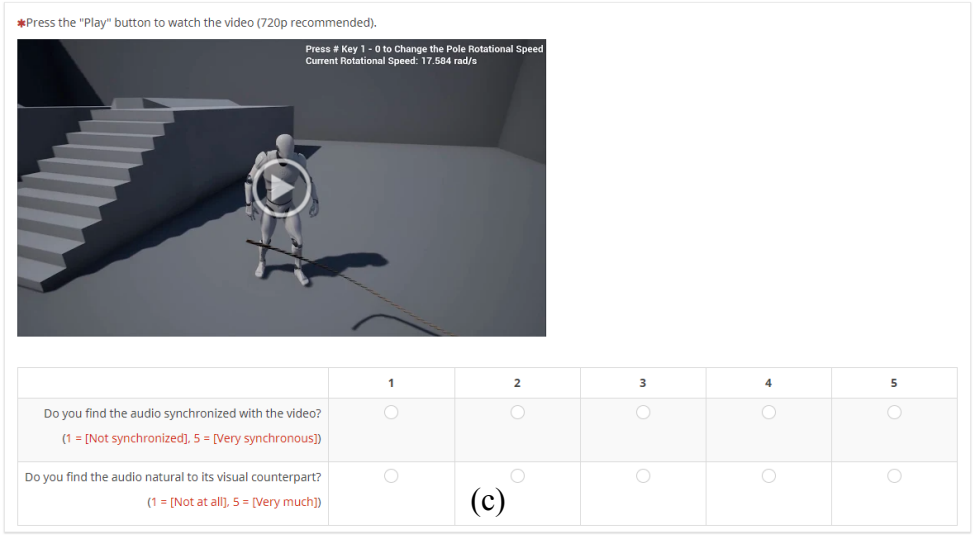


Figure 18 User study interface for our subjective evaluation. Study A (a) assessed recognisability, Study B (b) evaluated quality, Study C (c) verified synchronization.

5.2.2 Stimuli

The stimuli consisted of 35 recorded and 35 synthesized sequences of soft-body audio, as well as 46 video sequences of simulated soft-body interactions. A total of 12 categories of sounds were identified (Table 6, Group 1-3), where each class contained 2-4 audio/video clips. All synthesized sounds were generated using previous data-driven approaches. Specifically, the audio sequences of Group 1 were based on the results of Chapter 3, the sounds of Group 2 were based on the output from Chapter 4, and all of the Group 3 sounds were produced from the work of An et al. [52] (only the most relevant ones were selected). For a detailed description of each kind of sound, please refer to Section 3.3.1, Section 4.4.2, and the "EXAMPLE" section in the paper of An et al. [52]. As a comparison, we also recorded the corresponding real-world audio for each synthesized clip using an ultra-low-noise microphone in a sound isolation room (see Section 3.2.1), where we used real soft-body objects with the same sizes and materials as the virtual ones to closely replicate each type of interactions. To unify the sound quality, all of the audio signals were stored as 16-bit, stereo wave files with a sampling rate of 44.1 KHz (same as the output sound in Section 4.4.1).

5.2.3 Procedure

We conducted three online user studies[6] (see Appendix B for more detail) to evaluate the auditory perception of our synthesized soft-body sounds from three different perspectives. The entire study took approximately 60 to 90 minutes to finish, and all participants took part in all three experiments. The participants were required to wear a pair of headphone to listen to all sounds, and were allowed to take a break or withdraw at

[6] Due to the outbreak of the COVID-19 global pandemic, we were unable to carry out in-person user studies in a laboratory setting or a controlled environment.

any time during the study. To avoid learning effects and fatigue, the order of the first two studies (A and B) was randomized, while Study C would always be presented at the end as it contained important visual cues that might affect the consequences of the other audio-only studies.

Study A assessed the recognition rate of the audio, where the results were also compared to the IS from our objective metrics in Section 5.3.5. In this study, each trial included one audio clip and several images (one image for each audio category) with text labels (Figure 18 (a)). To facilitate straightforward assessment, we combined the selected 12 categories of sounds into three groups according to Table 6. In each group, participants were asked to identify which image best matched the given sound, and rate its realism on a 5-point Likert scale. To avoid confusion, we specifically explained to all participants that the "realism" here meant the presence of digital artifacts. A total of 70 audio sequences were presented in this study, among which 35 were real recordings and 35 were synthesized sounds. All audio clips and their related images were presented in random order, and the participants were unaware that this study included any procedurally-generated audio. At the beginning of the session, we also provided one real-world example (audio + image) for each sound category to help the participants practice and familiarize themselves with the sounds (see Appendix B).

Study B evaluated the quality of the synthesized sounds in comparison with their real-world recordings. This study was also used to examine the correlation between the perceived similarities of the two sounds and their objective FID scores from Section

5.3.5. In this experiment, two audio clips were presented side-by-side per trial: one from synthesized sound, and the other from its corresponding real recording. A related image with text label was also presented alongside the audio clips (Figure 18 (b)). We used the same stimuli as Study A, which consisted of 35 pairs of audio in total. For each pair of sounds, the participants were asked to rate which audio clip they preferred on a Likert scale, where 1 was labeled "*Strongly prefer audio clip 1*", 3 was "*No preference, audio 1&2 sound similar*", and 5 "*Strongly prefer audio clip 2*". This protocol creates an effective preference score for each audio pair, making it easier to be compared later to the FID score. The orders of synthesized and recorded sound in each trial, along with the trial order, were randomized. We emphasized to all participants that this study was to measure their opinions of the similarity of two sounds, and their preferences in terms of quality for different ones. Critically, participants were not told the two audio clips involved synthesized sound, nor how they related to each other. To establish a baseline for similarity, as well as avoid introducing biases in decision making, we set up three examples before the study began: one showing two clips that were very similar, one not so similar, and one that sounded very different (see Appendix B).

Finally, Study C was designed to verify the synchronization of the synthesized sounds and simulated motions, thus no recorded sound was included. In this study, we added visual cues (computer-simulated animations) to all of the audio stimuli in the previous two studies, and expanded the 12 categories of sounds in Table 6 by adding three more classes: *cloth_interc*, *cloth_drop*, and *rope_interc* from the results of Chapter 4. A series of 46 randomly-ordered video clips with synchronized procedural audio were presented

to the participants (Figure 18 (c)). During each trial, the participant was asked to detect any asynchrony between audio and visual, and rank their synchronization on a 5-point Likert scale. To provide further insight, participants also needed to assign a subjective score of 1-5 for criteria of whether they found the audio natural to its visual counterpart, where 1 was labeled "*Not at all*" and 5 "*Very much*". Again, to set up a baseline for synchronization, we provided three examples at the start of the session showing one video clip with perfectively synchronized audio, one with slightly unsynchronized audio, and one that was not synchronized at all (see Appendix B).

5.3 Objective Evaluation

Although using a perceptual user study is desirable for most of the sound evaluation applications, our ultimate goal is to develop a standardized and automatic method that can serve as an alternative to subjective evaluation. To this end, we propose two objective evaluation metrics using deep learning techniques to reassess the recognisability and quality of the aforementioned soft-body audio from a different perspective.

5.3.1 Audio Features

Building an appropriate feature representation has always been treated as the key process in any audio analysis task. When combined with deep neural networks like CNNs, such a feature extraction process can be performed jointly with objective optimization such as classification or evaluation. For decades, mel-frequency cepstral coefficients (MFCCs) have been used as the most common audio features in traditional audio signal processing. To calculate these coefficients, we usually apply mel filter banks to the magnitude spectra

of the audio, then take the logarithm of all filter bank energies and convert them with a discrete cosine transform. With deep neural networks, however, we can omit the last step since it would remove information and destroy spatial relations [62]. This leads to the log-mel spectrograms, which have become the dominant feature representation in deep learning. For our evaluation task, the log-mel spectrum will provide a more compact representation of the sound and require less data and training to achieve accurate results.

5.3.2 Deep Learning Models

After we represent the audio signals as feature vectors, they can be analyzed by various deep learning models. Similar to image processing, the depth of the neural network is crucial for the modeling capability. Deeper network typically yields higher accuracy, which is ideal to model complex temporal structures for audio classification or evaluation tasks. In this experiment, We used *GoogLeNet* (i.e. *Inception v1*) [96], a 22-layer deep CNN specifically designed for image recognition. This network has been trained on the *ImageNet* [83], and can classify images into 1000 categories with high accuracy. Moreover, as demonstrated by Boddapati et al. [74] and Hershey et al. [75], it also did well on audio classification tasks using mel spectrograms, which makes *GoogLeNet* a viable solution for analyzing features of soft-body audio.

5.3.3 Training Data

CNNs are known to be most beneficial when applied to large training datasets. For image processing, we have the *ImageNet* [83], a database consisting of more than 14 million hand-labeled images. Unfortunately, no such dataset exists in the audio domain for now.

To our knowledge, the largest dataset available for environmental sound classification is the *AudioSet* [97], which contains over 2 million human-labeled audio snippets. Therefore, to increase both the efficiency and training accuracy, most of the current works choose to use an image-based network to process sounds by converting their feature representations into visual images.

For this evaluation task, our training dataset included 135 pre-recorded soft-body audio clips (over 60 minutes in total), which were built upon the source databases used to generate the simulated sounds (see Section 3.2.1, Section 4.3.1, and An et al. [52] for detail). These audio sequences were labeled under the same 12 classes as we used in the subjective evaluation (Table 6). Since the size of our training data was too small for *GoogLeNet*, we applied data augmentation (which is a strategy that artificially increases the number of variations in the training dataset by adding modified versions of the original data) to generate additional training data and used transfer learning (which is a technique that applies knowledge of one type of problem to a different but related problem) to modify and fine-tune only the last few layers of the network.

However, *GoogLeNet* is initially trained on *ImageNet*, which means it will not transfer well to spectrograms, and our preliminary experiment shows that this network cannot converge on a solution if directly trained using our audio dataset. To address this problem, we first configured the network to classify environmental sounds with a relatively large audio dataset called *UrbanSound8K* [98]. This dataset is a collection of 8,732 labeled urban sound excerpts (over 27 hours) from 10 classes, and has similar

audio features (such as impulse and continuous structures) to our original training set. Moreover, Boddapati et al. [74] have demonstrated that the classification accuracy on the dataset can achieve 93% when trained on *GoogLeNet* with spectrograms. The reconfigured network would then be trained on our own data.

5.3.4 Experimental Setup

The first step of our experiment was to retrain *GoogLeNet* using *UrbanSound8K* to ensure that the network could work well with the features of mel spectrograms. As each audio file in the dataset contained short stereo audio (<= 4 s) of various sampling rates, we pre-processed the database and initialized all the audio to 16-bit, 44.1 KHz (same as our subjective study). Then, to provide better spatial information, we converted any given audio clip into a mono signal by adding together half the signal amplitudes of each channel. Next, we divided the signal into 500 ms segments with 50% overlap, where each segment inherited all the labels of its parent audio. This helped to reduce feature dimensions and provided additional data to capture all combinations of features. These 500-ms segments were transformed into 128-bin mel spectrograms with a window size of 2048 samples (46 ms) and a hop size of 1024 samples (23 ms). The resulting mel spectrograms were converted into logarithmic scale, which would give us a series of 128 × 20 bins that represented the features of the audio (Figure 19).

Once we had all the feature vectors, we saved them as 224 × 224 pixels RGB images to form the input to the network. To prevent the network from overfitting, we randomly translated the training images up to 10 pixels and scaled them up to 10% along the X and

Y axis. Lastly, we modified the *GoogLeNet* by replacing the last two layers (i.e. "*loss3-classifier*" and "*output*") with new layers adapted to our dataset. The training parameters were set to the following values: solver type – stochastic gradient descent with momentum; momentum – 0.9; training period – 8 epochs; initial learning rate – 0.01; learning rate schedule – piecewise; learning rate drop factor – 0.2; learning rate drop period – 2, and the training time took 12 hours on a single GPU (*NVIDIA GeForce GTX 1060*). All implementations were done using *MATLAB* with *Deep Learning Toolbox* [99].

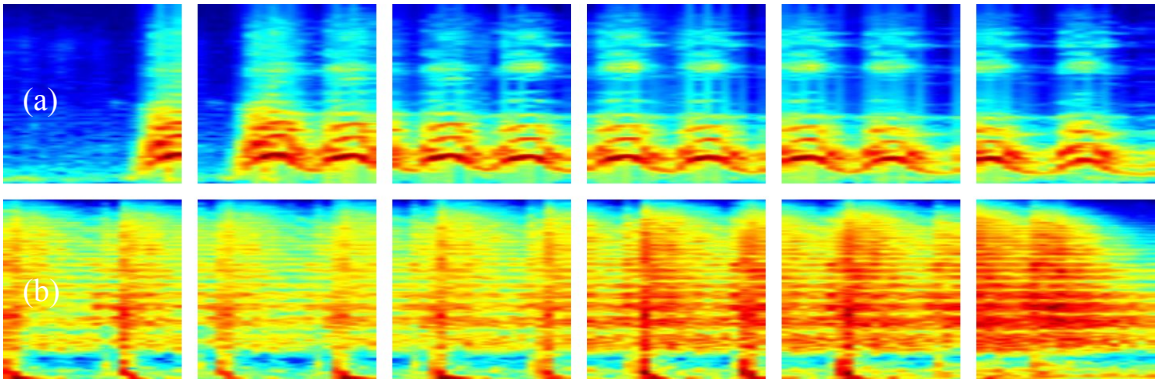


Figure 19 Examples of extracted features. (a) Dog bark, (b) gun shot.

We used 10-fold (with predefined splits/folds) cross validation to test the accuracy of the retrained network. Given a test fold consisted of n segments from m audio files, we first predicted the response of each individual segment, which would return an $n \times 10$ matrix, where the columns contained the estimation of a probability that a mel spectrogram segment contained the features of a particular audio class (10 in total). We then averaged these responses over each audio file, which would give us an $m \times 10$ matrix that contained the probability estimate of all the given audio. Next, we chose the maximum of the predictions for each audio as the predicted labels, and compared them to all the real

labels to calculate the overall classification accuracy. Finally, we repeated this process nine more times for other test folds, and the average accuracy we got was 88.36%.

5.3.5 Objective Metrics

We choose the IS and FID as our objective evaluation metrics to reassess all the three groups of sounds in Table 6. As these metrics are originally designed for image-generating GANs, we need to operate them on image-like spectrograms for audio.

Inception Score. Salimans et al. [69] proposed the IS, which used a pre-trained *Inception* classifier [84] to measure both the recognisability and diversity of generated images. Given the conditional label distribution of a generated image $p(y|\boldsymbol{x})$ with marginal distribution $p(y)$, the IS is defined as

$$IS(G) = exp(\mathbb{E}_{\boldsymbol{x}\sim p_g} D_{KL}(p(y|\boldsymbol{x}) \parallel p(y))) \tag{5.1}$$

where $\boldsymbol{x}\sim p_g$ is the image sampled from p_g, y is the set of labels and $p(y) = \int_x p(y|\boldsymbol{x})p_g(\boldsymbol{x})$, D_{KL} calculates the Kullback-Leibler (KL) divergence between the conditional and marginal probability distributions, and the results are exponentiated for easy comparison. The IS ranges from 1 to the number of classes of the classification model. With Equation (5.1), if a generated image contains distinctive and clear objects, it should be predicted confidently by the classifier and $p(y|\boldsymbol{x})$ should have a low entropy. Moreover, if the GAN model generates a high diversity of images, the marginal distribution $p(y)$ should have a high entropy. By combining these two, the IS would yield a high score, indicating that the generative model can produce diversified images with high recognisability.

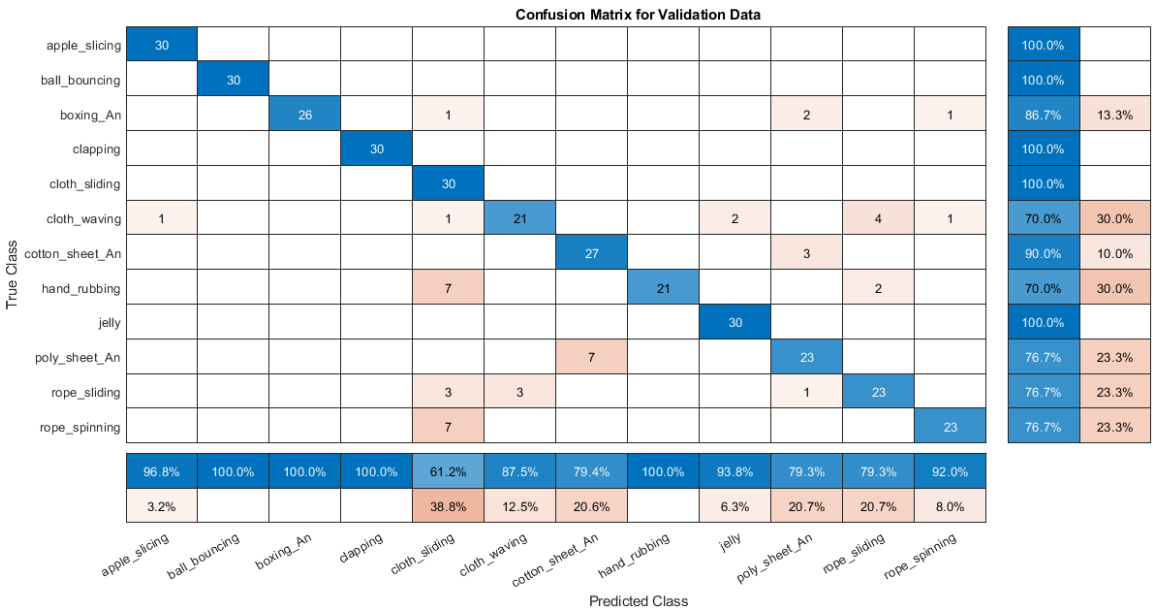


Figure 20 Confusion matrix for the validation data, where the values on the diagonal show the number of test audio files that were correctly classified in each category.

To calculate the IS of our audio, we first trained the modified *GoogLeNet* (*Inception v1*) on our soft-body datasets (Section 5.3.3) using the process and parameters illustrated in Section 5.3.4. Our new classifier contained 12 classes (Table 6), and could achieve an average accuracy of 87.50% (Figure 20) with 5-fold cross validation (as described in Section 5.3.4, but with 5 predefined folds due to the smaller dataset). For the next step, we used the same audio stimuli from our subjective evaluation (see Section 5.2.2), and segmented them into series of 100 ms overlapping frames. These segments were then converted into mel spectrograms images (224 × 224), and applied to the trained network to determine the conditional probability $p(y|\boldsymbol{x})$ for each image. As we did not measure diversity, we assumed that each class would be equally predicted so that the marginal probability $p(y)$ should be a uniform distribution. Finally, we calculated the KL divergence between the two for each image and averaged them over all images in each

class. The exponent of the results would give the IS that measured the recognisability of the audio.

Fréchet Inception Distance. To improve the IS, Heusel et al. [87] introduced FID, a metric that evaluates the quality of generated images by calculating the extracted feature distance between real and generated images. Given a set of synthesized images $\boldsymbol{x}$ and real images $\boldsymbol{r}$, we obtain their vision-relevant features from the coding layer of an *Inception* network. We then model the data distribution of these features using a multivariate Gaussian with mean and covariance ($\boldsymbol{\mu}$, $\boldsymbol{C}$) of the images, and the FID can be computed as the Fréchet distance (or Wasserstein-2 distance) between these two distributions:

$$FID(\boldsymbol{x}, \boldsymbol{r}) = \|\boldsymbol{\mu}_x - \boldsymbol{\mu}_r\|_2^2 + Tr(\boldsymbol{C}_x + \boldsymbol{C}_r - 2(\boldsymbol{C}_x\boldsymbol{C}_r)^{1/2}) \quad (5.2)$$

Here the $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_r$ refer to the feature-wise mean of the generated and real images, $\boldsymbol{C}_x$ and $\boldsymbol{C}_r$ are the covariance matrix of their feature vectors, and *Tr* sums up all the diagonal elements of the square matrix. The FID captures the similarity of generated images to real ones, and is more robust than IS. Lower score indicates that the two groups of images are more similar, and is minimized at 0 when these images are identical.

To measure the FID between our synthesized and recorded audio stimuli, the first step was again to retrain the modified *GoogLeNet* (*Inception v1*) to classify 12 new classes of soft-body sounds. Next, we transformed both the synthesized sounds and real recordings into sequences of mel spectrograms (same as above), and calculated their features from the last pooling layer (*global average pooling*) of the network. This layer had 1,024 activations, thus each image would be represented as a 1,024 feature vector. Finally, we

used Equation (5.2) to calculate the FID score between the two audio groups for each class. Figure 21 evaluates the computed FID with white Gaussian noise, showing that it correlates well with human judgment of sound quality.

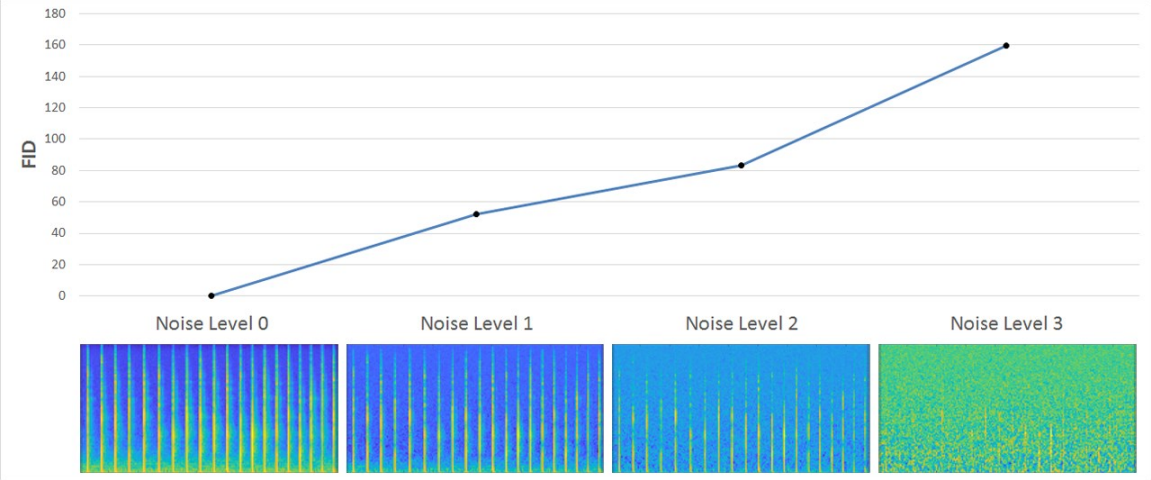


Figure 21 Evaluation of FID using clapping sound with increased level of noise. (Bottom) Mel spectrograms of clapping sounds, (Top) corresponding FID score chart.

5.4 Results

This section presents the results of our subjective and objective evaluation. For user study A, we show the cumulative recognition rates of each class of simulated and recorded sounds in Figure 22 (a), where the overall recognition accuracies for the simulated audio in Group 1, 2, and 3 are 71.50%, 59.43%, and 60.00% respectively, whereas for the recorded audio in each group, the recognition accuracies are 74.00%, 63.71%, and 60.80% respectively (Table 7). A more detailed recognition rate matrix of each class of sounds can be found in Appendix C.1. To seek more insight, Figure 22 (b) reflects the mean opinion scores (MOSs) (from 1 to 5, higher is better) for the realism of each type of sounds. The average scores for the simulated audio in Group 1-3 are 3.61 (SD = 1.17),

3.50 (SD = 1.00) and 3.35 (SD = 1.80), and the average scores for the recorded ones in each group are 3.71 (SD = 1.14), 3.48 (SD = 1.00), and 3.41 (SD = 1.08), respectively (Table 7). We also analyze these results with a within-subjects one-way analysis of variance (ANOVA) test, and find no statistically significant difference ($p > 0.05$) between simulated and recorded sounds for any of the audio category in all three groups (see Table 7, where the significance level was set to 0.05), which indicates that all of our simulated sounds convey as much expressiveness as the recorded ones.

Table 7 Statistics of user study A for each group of audio.

	Recognition Rate			**Realism**		
Audio Groups & Categories	Simulated	Recorded	ANOVA $F(1,48)$, p	Simulated (SD)	Recorded (SD)	ANOVA $F(1,48)$, p
Group 1 Avg.	**71.50%**	**74.00%**		**3.61 (1.17)**	**3.71 (1.14)**	
ball_bouncing	73.00%	71.00%	0.03, 0.870	3.76 (1.24)	4.12 (0.99)	1.75, 0.192
clapping	70.67%	72.00%	0.13, 0.724	3.79 (0.98)	3.77 (0.89)	0.00, 0.947
hand_rubbing	66.67%	74.67%	1.18, 0.283	3.57 (1.08)	3.88 (1.03)	1.72, 0.196
apple_slicing	74.67%	78.67%	0.73, 0.397	3.27 (1.23)	3.24 (1.20)	0.01, 0.928
jelly	72.00%	74.67%	0.01, 0.909	3.61 (1.23)	3.40 (1.35)	0.92, 0.343
Group 2 Avg.	**59.43%**	**63.71%**		**3.50(1.00)**	**3.48(1.00)**	
cloth_waving	64.00%	64.80%	0.04, 0.838	3.65 (1.06)	3.62 (1.00)	0.02, 0.882
cloth_sliding	57.60%	58.40%	0.30, 0.586	3.32 (0.95)	3.24 (1.02)	0.12, 0.729
rope_spinning	58.00%	80.00%	2.61, 0.113	3.54 (1.11)	3.82 (0.92)	1.27, 0.264
rope_sliding	54.00%	58.00%	0.21, 0.646	3.54 (0.81)	3.40 (0.90)	1.01, 0.321
Group 3 Avg.	**60.00%**	**60.80%**		**3.35 (1.08)**	**3.41 (1.08)**	
cotton_An	48.00%	52.00%	0.05, 0.832	3.48 (0.97)	3.50 (1.09)	0.01, 0.941
poly_An	70.00%	72.00%	0.47, 0.494	3.34 (1.21)	3.48 (1.01)	0.50, 0.484
boxing_An	64.00%	56.00%	2.01, 0.162	3.12 (1.01)	3.08 (1.15)	0.02, 0.897

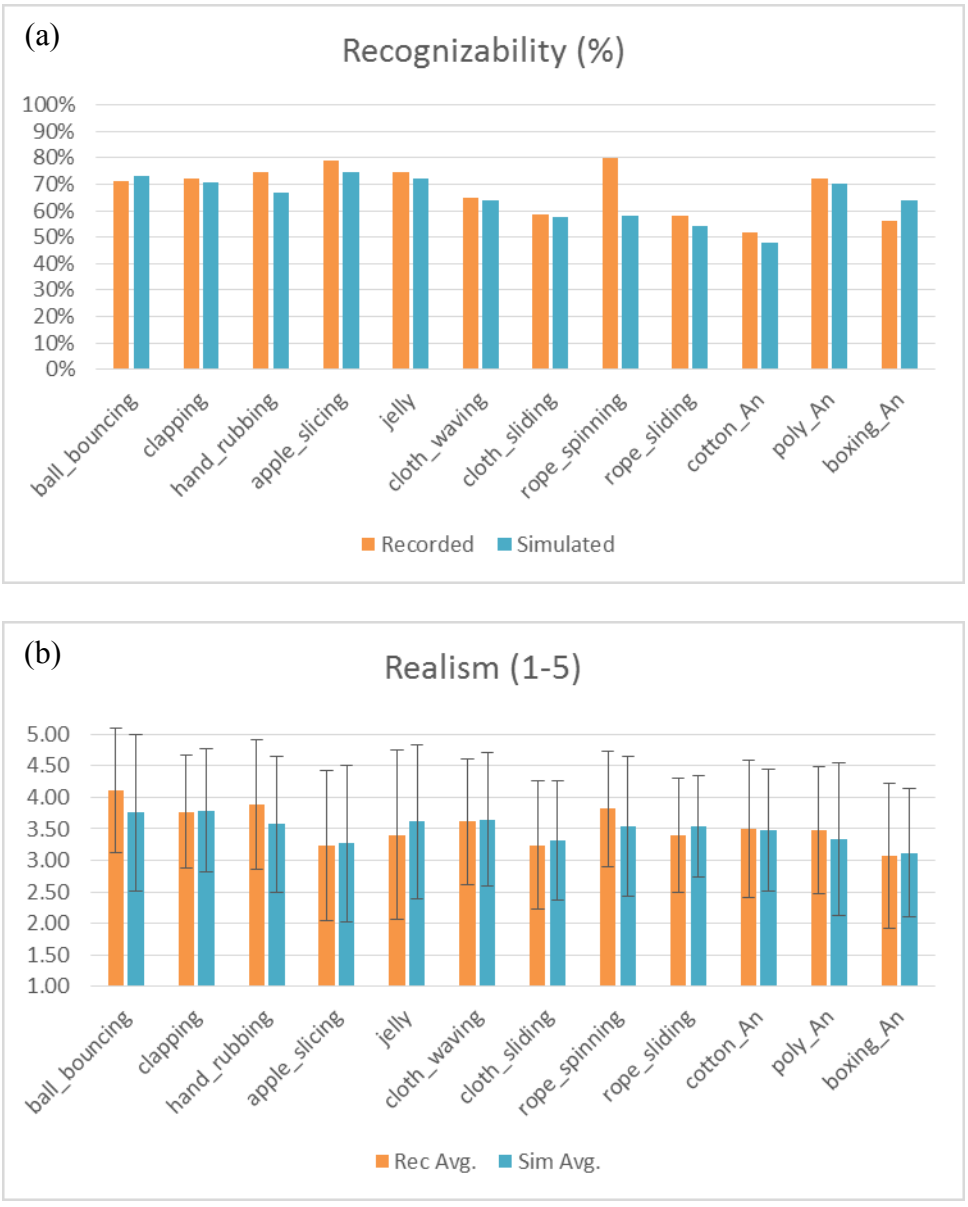


Figure 22 Results of user study A. (a) Recognition rates of simulated sounds compared to recorded ones. (b) MOSs for realism. The error bars here indicate the SDs.

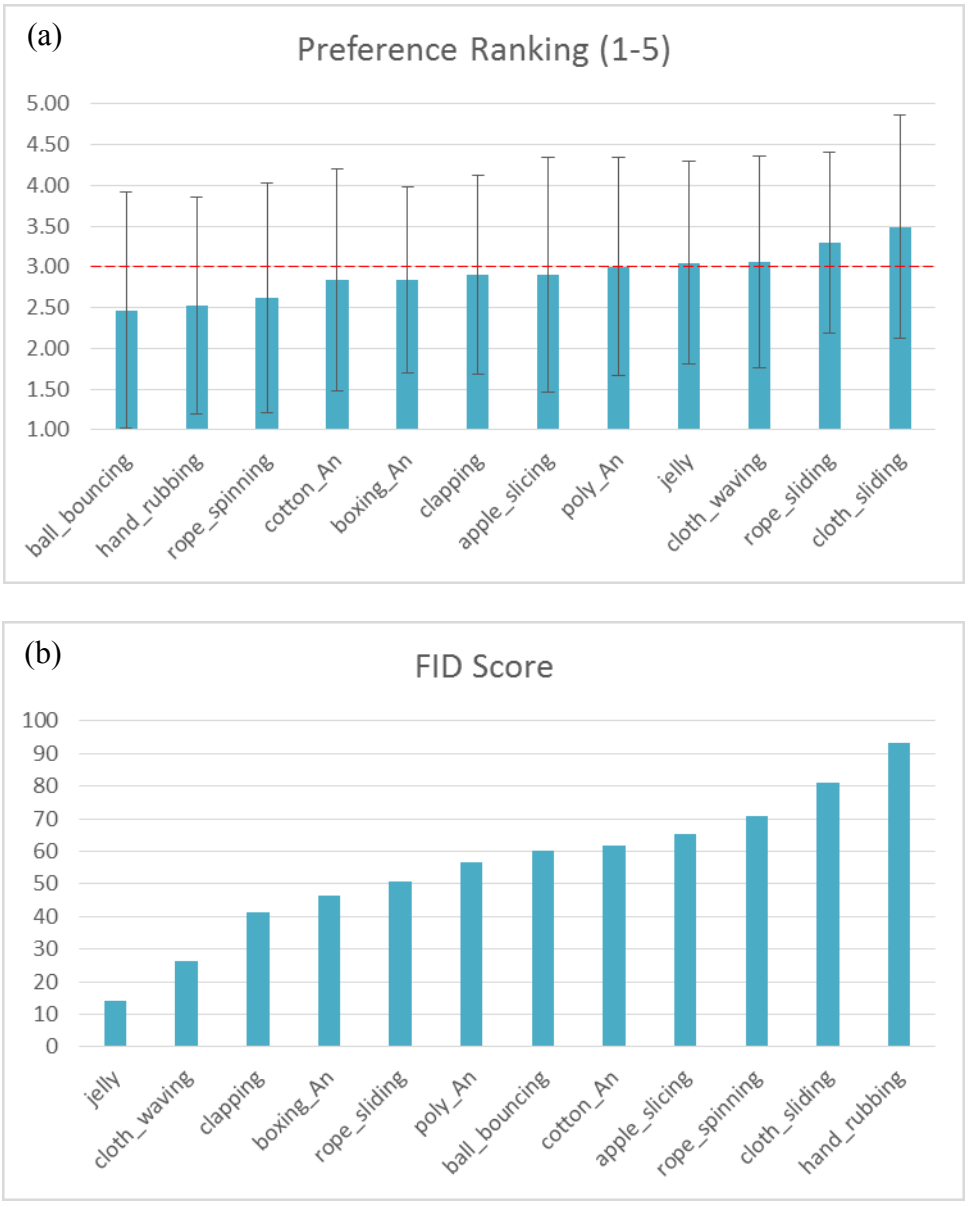


Figure 23 Results of user study B compared to FID. (a) MPSs sorted in ascending order, where the neutral score 3 is highlighted in red-dashed line. (b) FID scores sorted in ascending order.

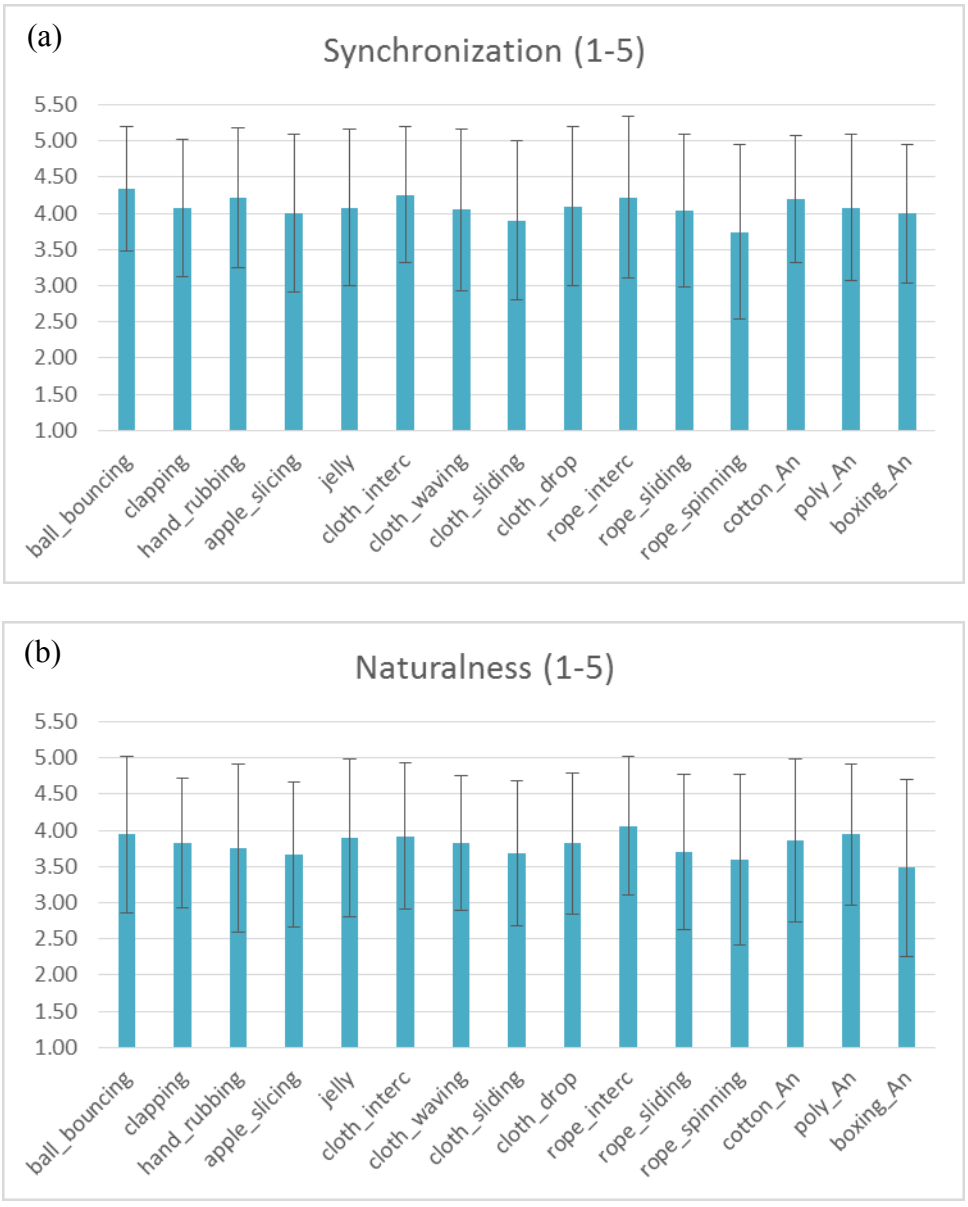


Figure 24 Results of user study C. (a) MOSs for synchronization between simulated sound and motion. (b) MOSs for naturalness of the sound to its vision. The error bars stand for SDs.

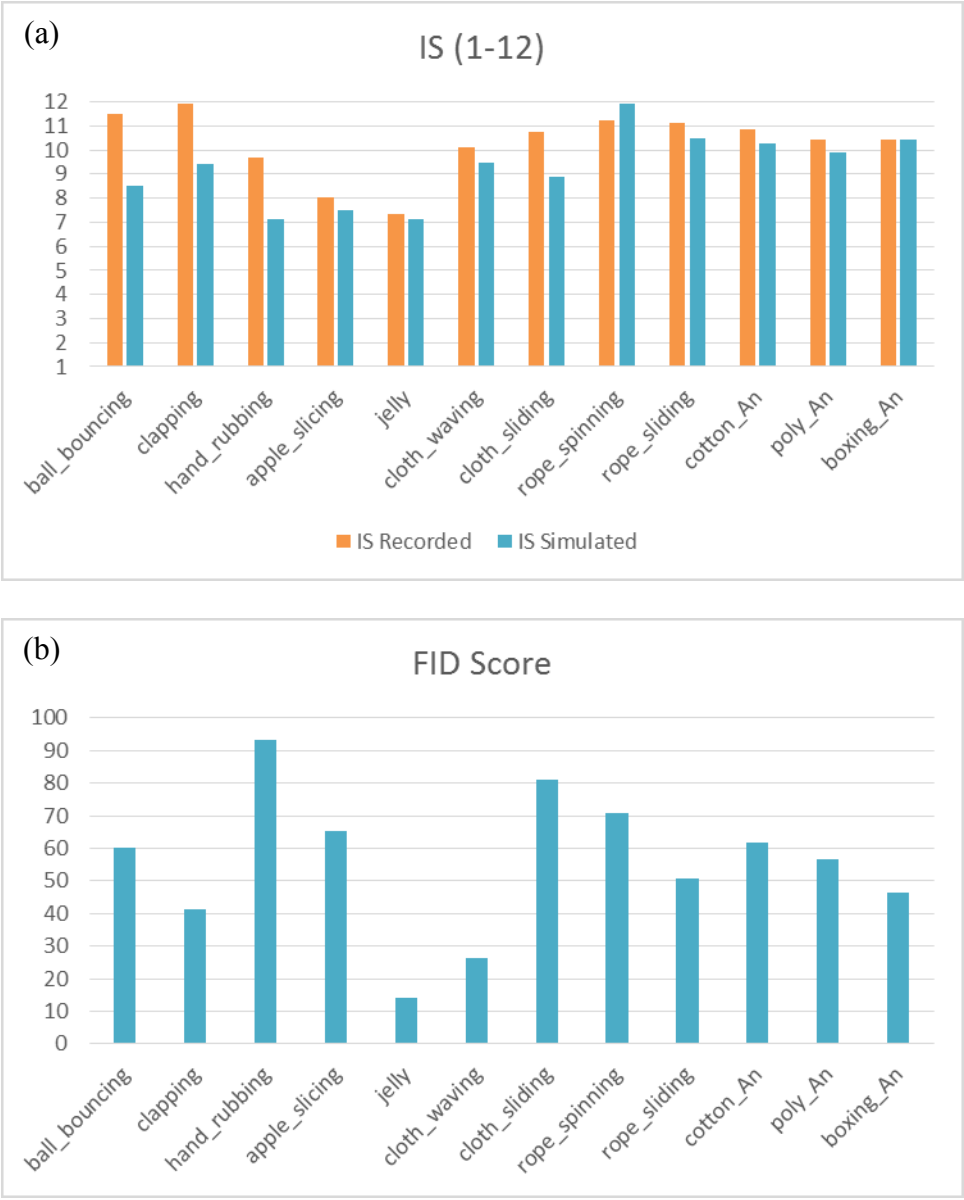


Figure 25 Results of objective metrics. (a) IS of both simulated and recorded audio. (b) FID scores between simulated and recorded audio.

Figure 23 (a) visualizes the results of user study B using a measure we called mean preference scores (MPSs) (from 1 to 5). Here score 1 refers to a strong preference for recorded audio, score 5 refers to a strong preference for simulated audio, and score 3 is a neutral score, meaning both audio clips are very similar and no preference for either clip.

These scores are sorted in ascending order for easy comparison, and the average scores over all classes are 2.75 (SD = 1.36) for audio group 1, 3.18 (SD = 1.34) for audio group 2, and 2.90 (SD = 1.30) for audio group 3, which indicates that generally our participants perceived all three groups of simulated sounds to be similar to the real recordings.

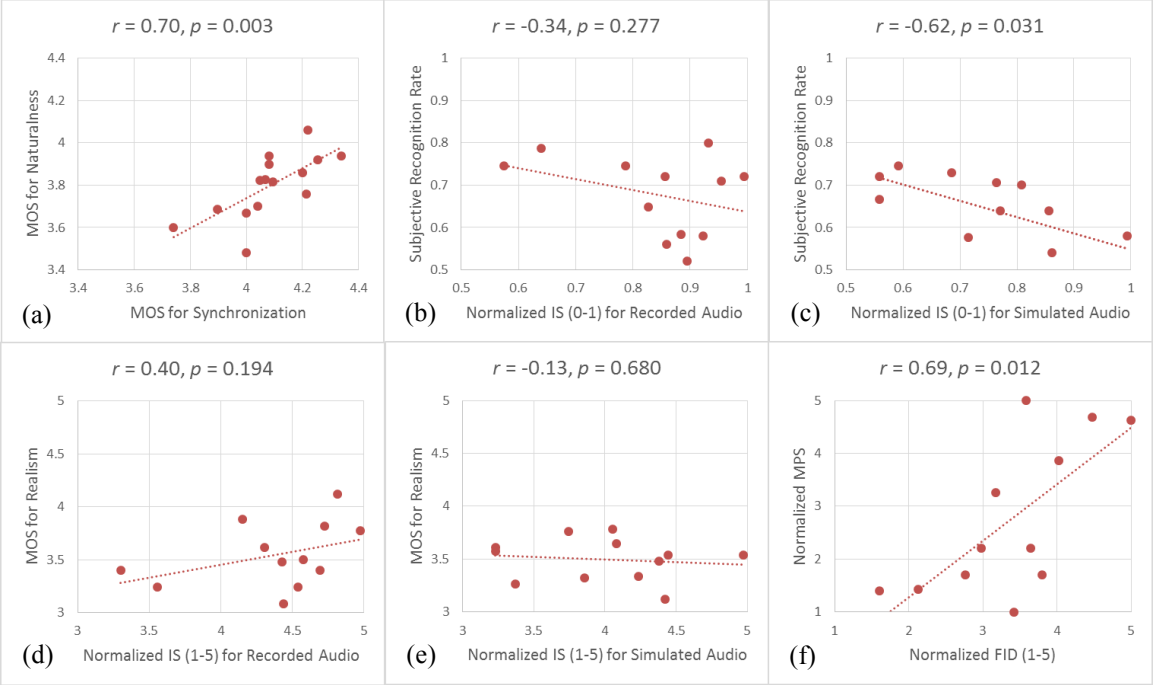


Figure 26 Scatter plots with trend lines that show the correlational analysis of (a) synchronization and naturalness, (b) IS and recognisability for recorded audio, (c) IS and recognisability for simulated audio, (d) IS and realism for recorded audio, (e) IS and realism for simulated audio, (f) FID and perceived similarity.

The results of user study C are presented in Figure 24, where we report the MOSs (1 to 5, higher is better) for both synchronization (a) and naturalness (b) of the simulated sound and motion. The average synchronization scores of all audio classes are calculated as 4.16 (SD = 0.98) for Group 1, 4.06 (SD = 1.08) for Group 2, and 4.11 (SD = 0.94) for

Group 3. The average scores of naturalness are 3.82 (SD = 1.05) for Group 1 audio, 3.81 (SD = 1.01) for Group 2 audio, and 3.82 (SD = 1.10) for Group 3 audio. Additionally, a correlation analysis is conducted on all the data (12 classes) of synchronization and naturalness, showing that they are strongly ($p < 0.05$) and positively ($0.5 < r < 1$) correlated to each other ($r = 0.70$, $p = 0.003$, see Figure 26 (a)). This suggests that the highly synchronized audio can improve the perceived realism of the sound to its vision.

The results for our objective evaluation appear in Figure 25. The IS (from 1 to 12, higher is better) is computed on both of the simulated and recorded sounds for each class (Figure 25 (a)), where the mean scores are 7.95 (SD = 1.00), 10.18 (SD = 1.34), and 10.20 (SD = 0.28) for simulated audio group 1,2, and 3, respectively, and 9.69 (SD = 2.04), 10.81 (SD = 0.52), and 10.57 (SD = 0.23) for recorded audio group 1-3. In addition, the one-way ANOVA for the IS of simulated and recorded sounds returns a statistically non-significant result ($F(1,22) = 3.10$, $p = 0.092$), suggesting that the measured recognisability of all simulated and recorded audio does not significantly differ from each other. To further analyze and compare the data, we normalize these results (IS) to fit within the range of our Study A data (see Appendix C.2). For recognisability, each IS (1-12) is normalized between 0-1 (recorded mean = 84.39% with SD = 12.47%, simulated mean = 75.06% with SD = 13.47%), and a correlational analysis is performed on these modified IS and our subjective recognition rates (Figure 22 (a)), which yields a negative ($-1 < r < 0$) correlation ($r = -0.34$, $p = 0.277$ for recorded data, and $r = -0.62$, $p = 0.031$ for simulated data, see Figure 26 (b) and (c)). Similarly, we normalize IS to the range 1-5 (recorded mean = 4.38 with SD = 0.50, simulated mean = 4.00 with SD = 0.54) to

compare the results of realism, and find a weak ($0 < r < 0.5$ or $-0.5 < r < 0$) and non-significant ($p > 0.05$) correlation between these two methods ($r = 0.40$, $p = 0.194$ for recorded, and $r = -0.13$, $p = 0.680$ for simulated, see Figure 26 (d) and (e)). This indicates that both evaluation criteria are largely independent of each other when assessing the recognisability and realism of the soft-body sounds.

Figure 25 (b) displays the FID scores (from 0 to infinity, lower is better) between the simulated and recorded sounds. The mean scores over all audio classes are 49.84 (SD = 28.96) for Group 1, 55.71 (SD = 26.47) for Group 2, and 54.86 (SD = 7.92) for Group 3. To compare these results with the data from Study B, we first normalize the FID scores for each class to fit within the range of 1 to 5 (mean = 3.38 with SD = 0.95, see Appendix C.2). Then, we convert the MPSs (Figure 23 (a)) by subtracting 3 (the neutral score) from each score and taking their absolute values. The results are also normalized to 1-5 (mean = 2.76 with SD = 1.45). These new scores will now measure the similarity between the two sounds, where lower score refers to higher similarity. Lastly, we conduct the correlation analysis on these scores, and find that there is a significant positive ($0.5 < r < 1$, $p < 0.05$) correlation between the FID and the subjective similarity scores ($r = 0.69$, $p = 0.012$, see Figure 26 (f)), which indicates that FID is consistent with participant's judgment of audio quality.

Finally, to compare the cloth sounds generated by our method and An et al.'s method, we use the audio/video tracks of the cloth-pole interactions in both methods (see the "Cloth with Pole" example in Chapter 4, and the "Cotton & Polyester Sheet" examples in [52]),

and specifically analyze the subjective evaluation results of these sounds, where all the average scores/rates and their SDs are shown in Table 8. We also conduct several one-way within-subjects ANOVAs to further test the statistical significances (alpha = 0.05) of these results, and find that although An et al.'s cloth sounds score better than ours in most cases (highlighted in Table 8), these differences are not at a significant level (all alpha > 0.05).

Table 8 Comparison of simulated cloth sounds. Our method vs. An et al.'s method.

	Cloth Type	Cotton Cloth	Polyester Cloth
Recognition Rate	Our method	**52.00%**	52.00%
	An et al.	36.00%	**64.00%**
	ANOVA	F=1.28, p=0.264	F=0.72, p=0.400
Realism Score (SD)	Our method	3.28 (0.94)	3.12 (1.17)
	An et al.	**3.56 (0.82)**	**3.44 (1.19)**
	ANOVA	F=1.26, p=0.266	F=0.92, p=0.342
Normalized Similarity Score (SD)	Our method	**3.08 (1.78)**	2.84 (1.82)
	An et al.	2.76 (1.76)	**3.16 (1.72)**
	ANOVA	F=0.41, p=0.526	F=0.41, p=0.526
Synchronization Score (SD)	Our method	4.00 (1.22)	3.72 (1.21)
	An et al.	**4.36 (0.86)**	**4.12 (0.97)**
	ANOVA	F=1.45, p=0.235	F=1.66, p=0.203
Naturalness Score (SD)	Our method	3.80 (1.04)	3.80 (1.04)
	An et al.	**3.88 (1.17)**	**4.00 (1.08)**
	ANOVA	F=0.07, p=0.799	F=0.44, p=0.508

5.5 Discussions

5.5.1 Recognisability

We find that the recognition rates and realism of our synthesized soft-body sounds compare favorably to recorded ones using both evaluation methods, however, no significant or positive correlation is found between the computational data (IS) and our human evaluation (Figure 26 (b)-(e)). From Table 7, we also see that generally both our synthesized elastic and plastic deformation sounds (Group 1&2) have obtained better or similar average accuracies (Group 1: 71.50%, Group 2: 59.43%, Group 3: 60.00%) and realism ratings (Group 1: 3.61, Group 2: 3.50, Group 3: 3.35) than the sounds from the work of An et al. (Group 3) albeit the fact that our contact sound models are much more simplified than their sound models. This demonstrates the effectiveness of our proposed sound models, as well as indicates that our participants tend to be more familiar with the sound of elastic deformations.

Additionally, it can be seen that the IS (normalized) achieves higher overall accuracies (recorded: 84.39% vs. 68.00%, simulated: 75.06% vs. 65.03%) and better scores (recorded: 4.38 vs. 3.57, simulated: 4.00 vs. 3.53) than human evaluation, which is most likely due to the fact that most of our human listeners have limited training and knowledge in VE audio so that they are not particularly good at recognizing complex soft-body sounds. For example, for both synthesized and recorded sounds, a number of participants have reported difficulty in differentiating between cotton and linen cloth, as well as rope and cloth sliding.

5.5.2 Quality

The perceived quality of our output soft-body audio is measured by comparing the similarity of synthesized sounds to their real recordings. Figure 23 shows the evaluated results side by side, where both MPSs (a) and FID (b) are sorted in ascending order. It appears that the majority of our impulsive/discrete sounds (such as jelly, cloth waving, and clapping, which have a low FID or a subjective score around 3) tend to be better at capturing the underlying features of the recorded audio, whereas most of our continuous sounds (such as cloth sliding, hand rubbing, and rope spinning, which have relatively high FID scores) tend to yield larger differences. This is possibly caused by the loss of temporal structures when resampling or concatenating continuous micro-sound units during synthesis.

In general, both simulated and recorded sounds are perceived similar to each other from our human evaluated results (MPS $\approx$ 3), and more importantly, we find a significant positive correlation between this subjective data and the obtained FID scores (Figure 26 (f)). However, for a number of classes including the ball bouncing and rope spinning, the participants still indicate a preference for recorded audio (MPS < 3). This is not surprising given that recorded sounds usually generate extra auditory clues that are missed from the synthesized sounds. However, it is interesting to note that our participants tend to prefer the simulated sounds in Group 2 over the recorded ones (average MPS = 3.18), especially for the sound of cloth and rope sliding, where their MPSs are close to 3.50. This implies that our sound generation method for plastic deformations has the ability to outperform traditional stored recordings for more complex

interactions such as sliding and rolling. Finally, we also notice that most of the MPSs have relatively high SDs, which suggests that individual's ability to perceive audio may vary significantly from each other (as was reported by van den Doel et al. [61]).

5.5.3 Synchronization

It is well known that poorly synchronized audio can disrupt the sense of immersion in VE [100]. User study C demonstrates that our procedurally-generated sounds have achieved a satisfactory level of synchronization that can reinforce the combined audio-visual experience and the naturalness of the soft-body interactions (Figure 26 (a)). However, simply adding synchronized audio to animations/motions does not necessarily mean that the perceived realism will improve, as the participant's perception of realism can also be influenced by other factors such as sound and visual fidelity. Although procedural soft-body audio offers fully-automated synchronization, it often involves a set of trade-offs in the output sound quality. Therefore, careful consideration must be taken during the synthesis process, which may lead to audio that is less synchronized but more perceptually plausible.

5.5.4 Comparison of Sound Synthesis Methods

Overall, our simulated sounds of cotton/polyester cloth convey similar subjective effectiveness as An et al.'s cloth sounds. From Table 8, we show that for both methods, the subjective scores/ratings are very close when we assess the recognisability, quality, as well as synchronization of the resulting sounds. This is further demonstrated by the results of our ANOVA tests, where no statistically significant difference is found

between any groups, indicating that our real-time synthesis approach is able to generate plausible cloth sounds that are comparable to previous off-line approach at a much lower cost.

As mentioned in Section 4.4.3, the synthesis method of An et al. contains a lengthy pre-computation process, which relies on a sound designer to provide a number of manual correspondences between the target and database audio, whereas our proposed method in Chapter 4 has sped up this process by using a number of simplified sound models to directly drive the synthesis of the database. Although our synthesis technique will often sacrifice some levels of details in the final audio, we have proved that it does not significantly affect the overall perceived realism of the sounds.

5.5.5 Comparison of Evaluation Criteria

While perceptual study is a useful and effective tool for subjective evaluation of soft-body audio, it is time-consuming and requires a large amount of human efforts. More importantly, it may not accurately reflect the perceived differences between sounds due to possible judgment biases of participants. In contrast, objective metrics are much faster, require fewer resources, and can generally produce more accurate results for performance-related evaluations. When combined with proper psychoacoustic principles, they can be used to efficiently capture and compare the characteristics of the target sounds.

The intuition behind the effectiveness of IS lies in its ability to recover good estimations of the conditional class distribution $p(y|\boldsymbol{x})$, as well as the marginal class distribution $p(y)$ of the generated images/sounds. However, our IS is calculated based on the assumption that each class of sounds would be produced with uniform probability. Consequently, the predicted marginal distribution may cast doubt on the actual performance of the score, and our experiments show that the resulting IS correlates poorly with the human perception. Furthermore, the accuracy of the IS can also be affected by the *Inception* network in many ways, such as overfitting and underfitting, and there are several other issues with this metric (discussed by Barratt and Sharma [101]) that can make it undesirable for evaluating the procedural soft-body audio.

Alternatively, FID uses the statistics of both synthesized and real world samples to assess the sound quality. With deep CNNs, this metric is able to effectively identify the subtle variations between the sounds and accurately measure the distance of their feature vectors. In general, FID is more robust to noise than the IS, and we find that it correlates well with the perceived similarities between the synthesized and recorded audio. Therefore, we can conclude that FID is a reasonable metric for the evaluation of soft-body sound quality, and can serve as a relatively quick and inexpensive ways to valid the performance of sound generation methods.

5.6 Summary

In this chapter, we have presented both subjective and objective methods to evaluate our synthesized audio from Chapters 3 & 4. The results demonstrate that our data-driven

approaches are able to generate a range of plausible soft-body sounds that are comparable to the recorded ones. We also find that while subjective evaluation is still a dominant method that offers many advantages, objective metrics show promise in performing quick and reliable assessments of the generated sounds. For IS, we show that it has several shortcomings and does not correlate well with our subjective data, thus this metric should only be used as a rough guide to validate the recognisability of the audio. On the other hand, we find that FID, although specifically designed for GANs, can also be effective in evaluating the perceptual quality of the simulated sounds.

Chapter 6: Conclusions and Extensions

6.1 Summary of Results

Generating realistic sounds for soft-body interactions is a challenging task and has been overlooked for decades. In this thesis, we aimed to close this gap by investigating practical and efficient synthesis methods that were able to automatically create audio for typical soft-body objects in computer graphics and interactive applications. Our research was carried out in three phases.

In research phase one, we proposed a novel data-driven approach to simultaneously synthesize audio based on the elastic deformations of soft bodies. Our method improves on existing granular synthesis technique by having both continuous and discrete contact models coupled with a novel and efficient control technique which retargets grains of real-world database recordings to match the motion reported from these contact models. This audio generation process is more computationally efficient than physically-based synthesis, and saves a large amount of effort over traditional Foley artistry. We demonstrated the effectiveness of this method on a variety of elastic deformations including a basketball bouncing, apple slicing, hand clapping/rubbing and a jelly simulation.

Research phase two explored the sound generation technique for plastic deformations, where we proposed a real-time, concatenate synthesis method for two most common plastically deformable models – cloth and rope. A number of simplified parametric sound

models are introduced, which can effectively analyze the contact states and deformations of different sounding objects at runtime. These analyzed data are then combined with a fast feature correspondence algorithm we created to select and synthesize segmented database sound units with added spatialization effects. Our approach does not require a lengthy pre-computation process compared to existing methods, and is compatible with any particle-based soft-body dynamics in video games. We implemented this method in *UE4* and have shown that it can produce a range of cloth and rope sounds with less memory and computing power consumption.

Finally, in phase three of our research, we investigated the evaluation of our generated sounds using both subjective and objective approaches. Our subjective evaluation consisted of a three-part perceptual study, where we examined the recognisability, quality, and synchronization of the synthesized audio. For objective evaluation, we adapted the metrics from GANs, and measured the recognisability as well as the quality of our sounds from a different angle. The results have shown that our proposed methods are able to produce high-quality audio that are seamlessly synchronized with both elastic and plastic deformations and conveying as much expressiveness as recorded sounds. Furthermore, our simplified sound models can drastically increase the synthesis speed without sacrificing the overall fidelity of the final audio. In addition, our results have indicated that objective metric such as FID tends to be a more efficient alternative to measure the output sound quality.

6.2 Limitations & Future Work

Procedural audio generation for soft bodies is a relatively new direction, and there are a number of limitations as well as many opportunities for future work.

Sound Models. Our current contact models and parametric sound models are built on particle-based soft-body simulators, which only take into accounts limited number of soft-body objects and cannot simulate more complex interactions such as melting, character clothing or hair physics. However, as our method only utilizes the information of each particle's global locations, which are usually accessible in most commercial off-the-shelf simulators, we can easily extend it to other soft-body systems that are dedicated to more specific types of objects/interactions.

Moreover, since our frictional contact model in Chapter 4 only calculates the total number of contact particles and their average sliding speeds, it cannot produce sounds for near-stationary contact events such as tablecloth slowly moving off table, which would generate near-zero average sliding speed. Similarly, we only consider a single aggregate buckling event energy for crumpling sounds, which cannot provide enough accuracy for highly complex interactions. To resolve these issues, a possible future direction is to use each moving contact/buckling particle separately as sound source, but this would generally lead to greater cost.

Source Database. Since we are generating sounds from pre-recorded databases, the range of the final synthesized sounds will depend on the diversity of the database.

Currently, we have only recorded sliding sounds between same type of materials, and our continuous/impulsive sounds as well as crumpling/aerodynamic models are only limited to certain type of motions. This makes it impossible to generate sounds from other sources of the interaction. For example, when we slide the cloth/rope with the pole, the frictions would also cause the pole itself to vibrate and produce sounds. When we hit the cloth with a large force, part of the cloth surface would momentarily act like a vibrating membrane that makes a loud impact noise. These effects can be observed in our supplemental video #2, but are missing in our synthesized results. Therefore, in the future we will expand our database to include more sounds from difference sources.

During the synthesis process, for simplicity we assume that all the sounds in the database obey the linear superposition principle, and can be added together. However, it is possible that in reality these sounds can exhibit certain non-linear characters from combinations of different interactions (such as sliding and stretching a rope at the same time), which may affect the final combined results. Future work will perform more experiments to explore such non-linearities.

Synthesis Parameters. The quality of the results will also be affected by the parameters we choose for synthesis. In Chapter 3, we only considered the grain peak amplitude and length to identify appropriate grains, and in Chapter 4, we only focused on the average sliding speed, total curvature energy change, as well as maximum Aeolian frequency to synthesize the cloth and rope sound. This may limit the type of interactions we can do for certain soft-body simulations. The real-world sound can also be influenced by other

factors such as object shape and size. Future work will explore more descriptive features to better model the sound.

Currently, all of our synthesis parameters are based on experimental or heuristic approaches, which only need to be specified once and can be reused. To further simplify this process, we plan to develop a dedicated interface with a set of presets that can suggest parameters for the best possible results and allow the users to customize what kind of sounds they expect to hear in the output.

Spatialization. Although our final sounds in Chapter 4 were spatialized according to the player's position, we did not address the issue of sound propagation. Instead, to minimize such effects, we simply used open environments with few obstacles for all of our examples. In general, numerical sound radiation for soft bodies is too computation-intensive for real-time applications, and many sound designers prefer to use alternative approaches to render virtual soundscape (such as ray-traced/wave-based sound propagation). This leads to another different area of research, which would be interesting to explore (such as Cowan and Kapralos [102]) and combine with our work in the future.

Evaluation. Though our evaluation focuses on the sound of soft bodies, we believe that these criteria can be extended to evaluate other types of audio such as rigid bodies and fluids. Our investigation mainly concerns two most widely used evaluation metrics for GANs, however, there could be other methods from the field of image processing (such as structural similarity (SSIM) index) that are more appropriate for soft-body sounds, and

we plan to investigate them in the future. Currently, the IS and FID are based on a modified *GoogLeNet* that is trained on *UrbanSound8K* and our own dataset. This network is limited to classifying only a handful types of sounds, which makes evaluating other classes of audio less consistent and thorough. Thus, future work will demand larger datasets as well as deeper CNNs dedicated to more categories of soft-body interactions. Finally, we would like to integrate the FID as part of the workflow for soft-body audio synthesis, so that the sound designer can easily check and tweak the quality of the generated sounds.

For other potential future works, we will consider including more soft-body objects as well as databases, and potentially integrating our system with commercial sound engines such as *Wwise* and *FMOD*. We are also interested in implementing our work with 3D audio, surround sound (such as *Dolby Atmos*), virtual reality and haptic technology, and see how these techniques would affect our perception in interacting with soft-body objects.

Appendices

Appendix A Taxonomy of Related Methods for Procedural Audio Generation

In Appendix A, we identify and categorize six groups of related methods for procedural sound synthesis, and discuss them in more detail. These methods are derived from previous works (see Chapter 2) that we find suitable as potential candidates for synthesizing soft-body audios. We will begin with a brief introduction to physically-based approach using modal synthesis, then move onto the concept of sound texture modeling, and finally focus more on data-driven approaches.

A.1 Modal Synthesis

As stated in Section 2.1.1, the standard framework to compute the sound generated by rigid bodies is modal synthesis. This method involves calculating the frequencies generated by the vibrating surface of the object, and computing the acoustic radiation caused by each frequency band [6]. Any object that exhibits a few modes and is excited by striking or plucking can be a candidate for modal modeling.

Modal Vibration [70]**.** To approximate a solid object's vibration, we use a linear vibration equation:

$$M\ddot{x} + D\dot{x} + Kx = f \tag{a.1}$$

where *M*, *K*, and *D* are respectively the mass, stiffness, and damping matrices depending on the object materials. $x \in R^{3n}$ describes the finite element nodal displacement with *n*

nodes, and $f \in R^{3n}$ is the external force driving the vibration. The damping matrix D is usually approximated using the Rayleigh damping model, which represents the damping matrix as a linear combination of mass matrix and stiffness matrix: $D = \alpha M + \beta K$, where the scalars α and β are user-specified parameters. Linear modal analysis then solves a generalized eigenvalue problem $KU = MUS$ to compute a modal shape matrix U and a diagonal eigenvalue matrix S. The former describes the vibration pattern of each mode while the latter indicates the square of undamped natural frequencies, i.e., $S_{i,i} = \omega^2$. Substituting $u = Uq$ and then pre-multiplying U on both sides of (a.1) decouples the system into a set of 1D second-order ordinary differential equations (ODEs), each of which is an ODE describing the modal vibration of a single mode i, namely,

$$\ddot{q}_\iota + d_i \dot{q}_\iota + \omega_i^2 q_i = U_i^T f \tag{a.2}$$

where d_i is the damping parameter of mode i, and U_i is the i-th column of U. The solution to this decoupled system (a.2) consists of a bank of modes, that is, damped sinusoidal waves. The i-th mode looks like:

$$q_i = a_i e^{-d_i t} \sin(2\pi f_i t + \theta_i) \tag{a.3}$$

where f_i is the frequency of the mode, d_i is the damping coefficient, a_i is the excited amplitude, and θ_i is the initial phase. The frequency, damping, and amplitude together define the feature φ of mode i: $\varphi_i = (f_i, d_i, a_i)$.

Sound Radiation [24]**.** A standard tool to model the sound radiation is the Helmholtz equation. This is the most expensive step of generating a modal sound. For every vibration mode with a frequency ω, it can be computed by solving a frequency-domain wave equation (Helmholtz equation),

$$\nabla^2 p(x) + k^2 p(x) = 0, x \in \Omega \quad \text{(a.4)}$$

where $p(x)$ is the acoustic transfer value at x, $k = \omega/c$ is the wavenumber of the corresponding vibration mode, and c is the speed of sound.

Frequency-Related Parameters [103]**.** Two sets of parameters are of particular importance for achieving desired sound characteristics: vibration frequency ω_i, which determines sound pitch, and damping coefficients, and d_i, which affects the timbre of particular materials. For example, a small damping value results in the long ringing sounds that metals often produce, whereas a large damping value tends to produce sounds more like wood. In standard modal sound models, the user can change material parameters such as Young's modulus E to adjust ω_i, and change α and β values in Rayleigh damping to control d_i.

Modal synthesis is popular because of its physically-based origins and plausibility when simulating rigid-body sounds. However, highly deformable bodies are typically poor candidates for linear modal synthesis. These objects tend to undergo large displacements and change shapes during the simulation. This causes vibration modes to change frequency over time, and breaks the assumption of near-rigid motion when computing acoustic radiation [104]. Therefore, simulating acoustic emissions from highly deformable objects is extremely expensive. Within the standard linear modal synthesis pipeline, correctly calculating eigenmodes of soft objects requires a fine internal tetrahedral mesh, which is memory-inefficient. Furthermore, large deformations can also

produce complex aeroacoustic effects. While it is possible to model this behavior using modal synthesis, it is much more efficiently captured with data-driven methods.

A.2 Sound Texture Modeling

The core of data-driven synthesis is to recreate new sound textures based on existing ones. Sound texture can be defined using two major constraints: constant long-term characteristics and attention span [105], [106]. A sound texture should exhibit similar characteristics over time. It can have local structure and randomness, but the characteristics of the fine structure must remain constant on the large scale. Attention span is the maximum time between events before they become distinct. High-level characteristics must be exposed within the attention span of a few seconds. Sound texture modeling allows for a more flexible playback of sound samples and is a widely used concept in computer music.

The generation of sound texture is at the intersection of many fields of research. Most of the published works pursue different kinds of analysis/re-synthesis approaches. Generally, they can be divided into methods that try to transfer existing methods from computer graphics (e.g. motion-driven sound synthesis, wavelet tree learning) and that take advantage of existing methods found in common computer music systems (e.g. granular synthesis) and speech synthesis (e.g. concatenative sound synthesis). Nevertheless, all approaches start from a given audio sample and share the question of how to perform the best segmentation, which parameters to extract and how to do the best

re-synthesis in order to create a new sample longer in duration but with similar quality to the original.

A.3 Motion-Driven Sound Synthesis

Motion-driven synthesis [49] provides a fully automatic control technique where a training motion signal is segmented and used to map sound onto a new motion signal. This method is inspired by the work of Bar-Joseph et al [107], where complex sounds are created by combining thousands of brief acoustical events. While the Bar-Joseph algorithm works well on both stochastic and periodic sound textures, it does not provide any control over the new instances of the sound texture it generates.

The motion-driven synthesis is a revised version of the Bar-Joseph algorithm, allowing user to control not only the location of the new synthesized sound, but also the transition between two different sound textures. This is achieved by enabling the user to manually specify what types of sounds from the input audio should occur when, and for how long, in the output sounds. These user-preferences translate into either soft or hard constraints during synthesis. The information in the animation's motion curves is also used to facilitate the process.

Additionally, this method is combined with an animation alignment algorithm, which allows these synthesis constraints to be determined semi/fully-automatically by matching similar motion events from a source animation to the target animation. As a result, the user only needs to provide a source animation and its associated sound segments. Given a

different target animation of the same nature, the algorithm finds the closest matches in the source motion to the target motion, and assigns the matches' associated sound events as constraints to the synthesis of the target animation's new soundtrack.

An advantage of this method is that it provides a very natural way to specify soundtracks with broad user specifications. Instead of creating a soundtrack from scratch, a user can simply supply a sample animation with soundtrack, and automatically get desired sounds for new animations. This significantly simplifies existing soundtrack recycling. Unfortunately, this method is only valid for input candidates that allow themselves to be manually re-arranged without incurring perceptual problems (such as background noise). It does not work for sounds with clear progressions (such as an increasing siren sound). Moreover, since the method uses a low-dimensional motion signal to drive the synthesis, it suffers from obvious low-frequency artifacts.

A.4 Wavelet Tree Learning

The fundamental idea behind wavelets is to analyze the signal's local frequency at all scales and times, using a fast invertible hierarchical transform. A wavelet representation is a multiscale decomposition of the signal. We can view it as a complete tree, where each level stores the projections of the signal onto the wavelet basis functions of a certain resolution. All basis functions at a particular level are translated copies of a single function. Thus, the wavelet transform is a series of coefficients describing the behavior of the signal at dyadic scales and locations.

Wavelet trees can be adapted to model sound textures and audio backgrounds. To synthesize sounds that are similar to the original, we can use a statistical learning algorithm [108]. An input sound file is decomposed into wavelets. Out of the wavelets, the algorithm captures the joint statistics of the coefficients across time and scale. Then a multiple resolutions analysis tree is created that is used to create new collections of sound grains that have a similarity to the original sample sound. In the re-synthesis step, the inverse wavelet-transform is applied to obtain an output tree. By random granular combination, the texture sounds are resynthesized.

Wavelet tree learning is capable of generating stochastic, quasi-periodic, as well as mixed periodic and stochastic sound textures. However, it offers no automatic input control needed for general procedural sound synthesis. Similar to motion-driven synthesis, this method also has limited capabilities in terms of capturing long progressive sound phenomena, such as an accelerating car that takes several seconds.

A.5 Granular Synthesis

Granular synthesis is a classic method in computer music for resynthesizing sound textures. More recently, it has become a popular technique to create sounds for computer animation or other digital media. Generally speaking, all methods that start from an input sample use a special form of granular synthesis in their final synthesis step. Short segments of sounds are manipulated to form a sequence of audio signals that sound like a particular object or event. As an example, stochastic sound models and granular synthesis are often used to produce noise-like fluid sounds such as waterfalls [6]. Roads [50] gave

an excellent review on the theories and implementations of sound synthesis with this approach.

A sound grain is a brief microacoustic event, with a duration near the threshold of human auditory perception. Each grain contains a waveform shaped by an amplitude envelope. This waveform can be a fixed sine wave, or a single period extracted from a recorded audio. To create the unique sound of any object, we need to manipulate the parameters of the grains, which usually include envelope shape, duration, waveform, frequency band, density and fill factor. With these many parameters, it requires granular synthesis to have a massive amount of control data. Therefore, a global unit of organization (spatialization) is necessary for practical work, which requires the sound designer to specify the sound in global terms, while the granular synthesis algorithm fills in the details. This greatly reduces the amount of data that the sound designer must supply.

Depending on how we organize the grains, granular synthesis can have many types of forms, among which we find that the physical modeling (PhM) and sample sound granulation would serve as good candidates for synthesizing soft-body sounds. PhM starts from a mathematical description of the mechanical and acoustical behavior of a sounding object. Such granular process can be taken still further, through more abstract algorithms such as chaotic functions to set up a perceptually compelling mapping between chaotic behavior and the synthesis parameters. Another powerful means of sound transformation is the granulation of sampled sounds, where a sound signal is segmented into grains, modified in some way, and then reassembled in a new time order.

This may take the form of a continuous stream or a statistical cloud of sampled grains, but the exact manner in which granulation occurs will vary from implementation to implementation. Granulation is a purely time-domain operation, which stands in contrast to techniques such as the Fourier or wavelet transforms where a signal is analyzed in both time and frequency domains.

To sum up, granular synthesis offers unique opportunities to sound designers and suggests new ways of organizing sound textures. It is ideal for representing statistical processes of timbre evolution, but has not yet proven to be effective for generating more natural and smooth-sounding results. To move towards synthesizing soft-body sounds, we must utilize statistics derived from existing natural sound examples to construct new desired sounds. Note also that granular synthesis is a generative technique that lacks analysis methods. It should be used during the last step when combined with other analysis-synthesis techniques (such as motion-driven synthesis).

A.6 Concatenative Sound Synthesis

CSS [53] aims at generating a meaningful macroscopic waveform structure from a large number of shorter waveforms. It typically uses a database of segmented source sounds and a unit selection algorithm to find the units that best match the target sound. CSS has its origin in speech synthesis, where the sound quality requirements dictate the use of data-driven approach to represent each phoneme. It has also been adapted for musical instrument synthesis, where the target signal can be symbolically represented as a sequence musical notes.

Database Segmentation. The database holds references to the original sound files and analyzed data files. Before included in the database, the source sounds need to be time-segmented into units. This can be done in various ways: by hand, by an external segmentation program, or by alignment of the pitch contour, using a dynamic time-warping algorithm. Information contained in the sound, such as MIDI note number, is usually attached to each segmented unit. This symbolic data can later be exploited for unit selection. Additionally, a sub-segmentation can be performed to further divide each sound segment into an attack phase, followed by a sustain phase, and a release phase. The release phase and the attack phase of the following unit together form a transition segment.

Analysis and Features. The segmented source sounds can be analyzed with standard signal processing methods, which typically analyze the signal's pitch, energy, spectrum, additive partials, and spectral envelope. These analysis data are then used to calculate the scalar features that describe each unit. Features are characteristics of the source signals. Generally, there are three classes of features: continuous, discrete, and symbolic. Continuous features are calculated from the analysis data over the duration of one unit. Discrete features take one numeric value per unit. Symbolic features take one symbolic value per unit, given by or derived from the symbolic score.

Target Specification. The target features can come from a symbolic score, such as a MIDI file, which contains discrete note values and other control parameters. They can

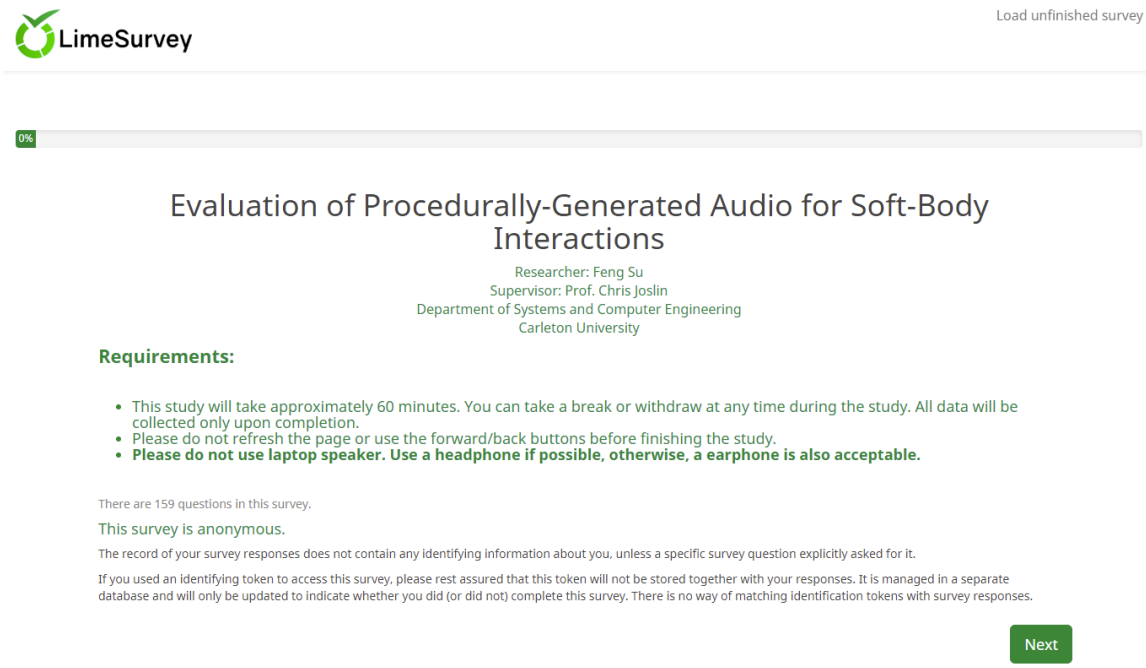
also come from an audio score, such as the frequency and envelope of a recording. To be accessible by the unit selection algorithm, the target specifications also need to be segmented into units, along with their corresponding features generated. Just like the source units, each resulting target unit can then be sub-segmented into attack, sustain, and release phase. This gives the unit selection algorithm more freedom to combine transitions and sustained parts.

Unit Selection & Synthesis. The unit selection algorithm incrementally finds the units from the database that best match the given synthesis target units. The selection is performed according to the features of the units. The selected units are then transformed to fully match the target specifications, and concatenated together. If the database is sufficiently large, the probability for an exact matching unit to be found is high, thus the need to apply transformations can be reduced. Otherwise, transformation can be done to affect the unit's fundamental frequency (through resampling), energy (through multiplication), or spectral envelope (through filtering). For concatenation, all the selected units' segments are joined together with a slight overlap, and a crossfade of the overlapping part is performed.

CSS takes advantage of the information contained in the actual recordings. It offers high level control over generating new instances of sound textures and is more flexible in terms of resynthesizing existing sound units. However, the quality of CSS depends on the quality and variety of the database. As an example in speech synthesis, simply gluing all the phonemes together does not make for a realistic sound of the actual word. To

accurately capture the speech, we also need to record transitions between phonemes (called diphones). This requires much more storage, but the increase in quality is significant.

Appendix B Online Survey for Audio Evaluation



LimeSurvey

0%

Informed Consent Form (Please Read)

Name and Contact Information of Researchers:

- Feng Su, Carleton University, Department of Systems and Computer Engineering
- Tel.: 613-261-1708
- Email: fengsu@cmail.carleton.ca
- Supervisor and Contact Information: Prof. Chris Joslin, chrisjoslin@cunet.carleton.ca

Project Title

- Evaluation of Procedurally-Generated Audio for Soft-Body Interactions

Project Sponsor and Funder (if any)

- N/A

Carleton University Project Clearance

- CUREB-B Clearance #: **112523** Date of Clearance: **April 1, 2020**

***Invitation**

You are invited to take part in a research project. The information in this form is intended to help you understand what we are asking of you so that you can decide whether you agree to participate in this study. Your participation in this study is voluntary, and a decision not to participate will not be used against you in any way. As you read this form, and decide whether to participate, please ask all the questions you might have, take whatever time you need, and consult with others as you wish.

What is the purpose of the study?

This study aims at evaluating the subjective expressiveness of procedurally-generated audio for soft-body interactions in computer animations and games. Specifically, we are looking to explore: (a) the recognizability of synthesized sounds; (b) the quality of synthesized sounds compared to real-world recordings; and (c) the synchronization of sound and motion.

What will I be asked to do?

If you agree to take part in the study, we will ask you to complete a three-part online survey which will take approximately 60 minutes. For each part of the survey, please use a headphone/earphone and follow the instructions closely on the screen. Before the study, you will need to answer a few basic questions about yourself to help us collect some demographic information about our participants. In Study A, you will listen to a series of audio clips, and for each audio clip, you will answer a few questions based on your own opinion. In Study B, you will listen to a series of audio pairs. For each pair of audio clips, you will also answer some questions based on what you just hear. All of the audio clips/pairs will appear in random order, and you will not be told how they relate to each other. In Study C, you will watch a series of randomly organized video clips, again we will ask you to answer a few questions based on what you see. You can finish the entire survey in one sitting, or you can take a break between each session. We will not audio/video record or photograph any part of the test session.

Risks and Inconveniences

We do not anticipate any risks to participating in this study.

Possible Benefits

You may not receive any direct benefit from your participation in this study. However, your participation may allow sound designers to better understand what sound properties are important to capture, and what improvement can be made to enhance the quality of the procedural soft-body audio.

Compensation/Incentives

There will be a $20.00 Tim Hortons e-gift card as compensation for your time.

No waiver of your rights

By agreeing to this form, you are not waiving any rights or releasing the researchers from any liability.

Withdrawing from the study

If you withdraw your consent during the course of the study, all information collected from you before your withdrawal will be discarded.

After the study, you may request that your data be removed from the study and deleted by notice given to the Principal Investigator (named above) within 24 hours after your completion.

Confidentiality

We will remove all identifying information from the study data as soon as possible, which will be after 24 hours upon your completion of the study.

We will treat your personal information as confidential, although absolute privacy cannot be guaranteed. No information that discloses your identity will be released or published without your specific consent. Research records may be accessed by the Carleton University Research Ethics Board in order to ensure continuing ethics compliance.

The results of this study may be published or presented at an academic conference or journal, but the data will be presented so that it will not be possible to identify any participants unless you give your express consent.

All collected data will remain anonymous, and will be password-protected and temporarily stored in a private domain at LimeSurvey.com. After we receive all the responses, the private domain will be closed, and all the data will be encrypted and exported to a password-protected file on a secure computer. Research data will only be accessible by the researcher team..

Data Retention

After the study is completed, your de-identified data will be retained for future research use.

New information during the study

In the event that any changes could affect your decision to continue participating in this study, you will be promptly informed.

Ethics review

This project was reviewed and cleared by the Carleton University Research Ethics Board B. If you have any ethical concerns with the study, please contact Carleton University Research Ethics Board-B (by phone at 613-520-2600 ext. 4085 or by email at ethics@carleton.ca).

Statement of consent:

Choose one of the following answers

- ( ) By clicking "Next", you consent to participate in the research study as described above.

This is a question help text.

LimeSurvey

0%

Basic Information

*Please indicate your age:

Your answer must be between 18 and 100
Only an integer value may be entered in this field.

*Please specify your gender:

Choose one of the following answers

- Male
- Female
- Other

*How are you familiar with virtual environment?

1 = [No experience at all], 5 = [Very familiar]

1 2 3 4 5

*Please provide your preferred email address to receive your compensation ($20 Tim Hortons e-gift card) after you submit your response:

LimeSurvey

0%

Study A

Description

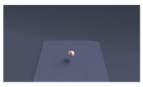
In this study, you will be presented with a series of audio clips and several images with text labels. For each audio clip, you will be asked to select which image best matches the sound, and rate how realistic the sound is.

Instructions:

- In each of the following pages:
- Please click the "Play" button to listen to the sound.
- You can play the sound as many times as you like.
- As soon as you are able, please read the questions and choose your responses by clicking the radio button below the answer.
- Press the "Next" button for the next page.

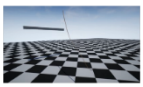
*Here are some example audio clips and their corresponding images:

Basketball on a carpet

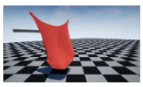


Jelly on a plate

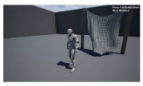
Sisal rope



Cotton sheet



Polyester sheet in the wind



Rope spinning

Choose one of the following answers

Please press the "Next" button to proceed.

LimeSurvey

0%

Study A

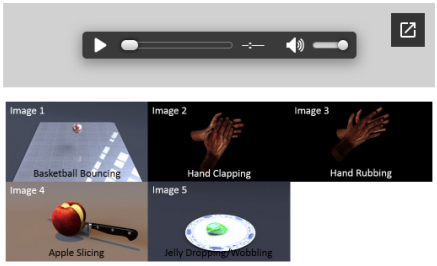
Description

In this study, you will be presented with a series of audio clips and several images with text labels. For each audio clip, you will be asked to select which image best matches the sound, and rate how realistic the sound is.

Instructions:

- In each of the following pages:
- Please click the "Play" button to listen to the sound.
- You can play the sound as many times as you like.
- As soon as you are able, please read the questions and choose your responses by clicking the radio button below the answer.
- Press the "Next" button for the next page.

*Please press the "Play" button to listen to the audio.



	1	2	3	4	5
Which image best matched the sound you just heard? (1 = [Image 1], 5 = [Image 5])	○	○	○	○	○
How realistic do you find the sound? (1 = [Not realistic], 5 = [Very realistic])	○	○	○	○	○

LimeSurvey

0%

Study B

Description

In this study, you will be presented with a series of related audio clip pairs and their corresponding images with text labels. For each pair of audio, you will be asked to rate which audio clip you prefer on a Likert scale.

Instructions:

- In each of the following pages:
- Please click the "Play" button to listen to both audio clip 1 and 2.
- You can play both audio clips as many times as you like.
- As soon as you are able, please read the questions and choose your responses by clicking the radio button below the answer.
- Press the "Next" button for the next page.

*Here are some example audio pairs:

Audio pair that are very similar

Audio 1:

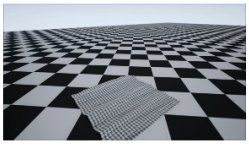
Audio 2:

Audio pair that are not so similar

Audio 1:

Audio 2:

Audio pair that are very different



Audio 1:

Audio 2:

Choose one of the following answers

Please press the "Next" button to proceed.

LimeSurvey

0%

Study B

Description

In this study, you will be presented with a series of related audio clip pairs and their corresponding images with text labels. For each pair of audio, you will be asked to rate which audio clip you prefer on a Likert scale.

Instructions:

- In each of the following pages:
- Please click the "Play" button to listen to both audio clip 1 and 2.
- You can play both audio clips as many times as you like.
- As soon as you are able, please read the questions and choose your responses by clicking the radio button below the answer.
- Press the "Next" button for the next page.

*

Audio 1:

Audio 2:

Apple Slicing:

Audio clip 1 and 2 are both related to the image above.

	1	2	3	4	5
Which audio clip do you prefer? (1 = [Strongly prefer audio clip 1], 3 = [No preference, audio 1&2 sound similar], 5 = [Strongly prefer audio clip 2])	○	○	○	○	○

LimeSurvey

0%

Study C

Description

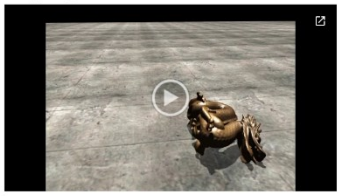
In this study, you will be presented with a series of video clips with synchronized audio clips. For each video clip, you will be asked to detect any asynchrony between audio and visuals, and rate how natural you find the sounds are.

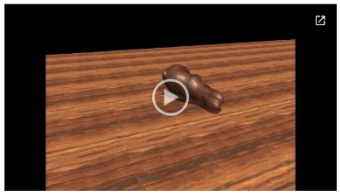
Instructions:

- In each of the following pages:
- Please click the "Play" button to watch the video.
- You can play the video as many times as you like.
- As soon as you are able, please read the questions and choose your responses by clicking the radio button below the answer.
- Press the "Next" button for the next page.

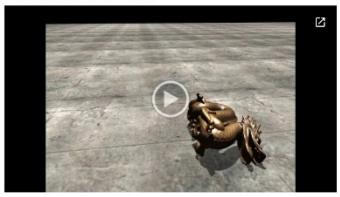
*Here are some example video clips:

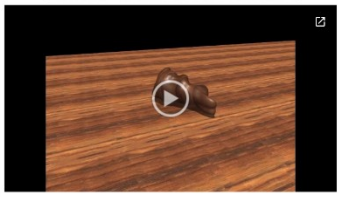
Video clip that is synchronized



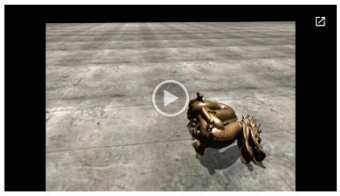


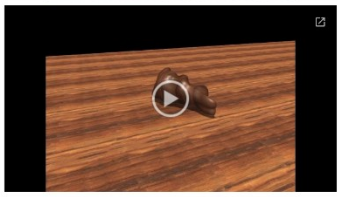
Video clip that is slightly unsynchronized





Video clip that is not synchronized at all





Choose one of the following answers

Please press the "Next" button to proceed.

LimeSurvey

0%

Study C

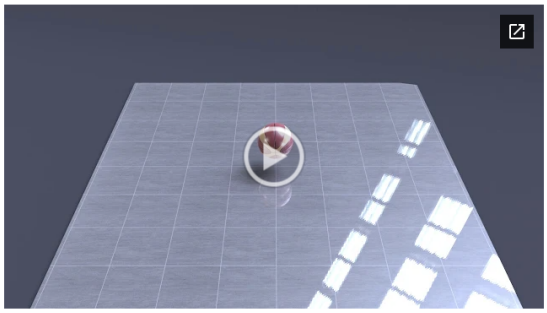
Description

In this study, you will be presented with a series of video clips with synchronized audio clips. For each video clip, you will be asked to detect any asynchrony between audio and visuals, and rate how natural you find the sounds are.

Instructions:

- In each of the following pages:
- Please click the "Play" button to watch the video.
- You can play the video as many times as you like.
- As soon as you are able, please read the questions and choose your responses by clicking the radio button below the answer.
- Press the "Next" button for the next page.

*Press the "Play" button to watch the video (720p recommended).



	1	2	3	4	5
Do you find the audio synchronized with the video? (1 = [Not synchronized], 5 = [Very synchronous])	○	○	○	○	○
Do you find the audio natural to its visual counterpart? (1 = [Not at all], 5 = [Very much])	○	○	○	○	○

Appendix C Statistics of Subjective and Objective Evaluation

C.1 Recognition Rate Matrix of Study A

Recorded Sounds:

Group 1	ball_bouncing	clapping	hand_rubbing	apple_slicing	jelly
ball_bouncing	**71.00%**	11.00%	8.00%	5.00%	5.00%
clapping	4.00%	**72.00%**	12.00%	8.00%	4.00%
hand_rubbing	1.33%	8.00%	**74.67%**	12.00%	4.00%
apple_slicing	2.67%	1.33%	16.00%	**78.67%**	1.33%
jelly	5.33%	5.33%	8.00%	6.67%	**74.67%**

Group 2	cloth_waving	rope_spinning	cloth_sliding	rope_sliding
cloth_waving	**64.80%**	9.60%	19.20%	6.40%
rope_spinning	2.00%	**80.00%**	14.00%	4.00%
cloth_sliding	4.00%	14.40%	**58.40%**	23.20%
rope_sliding	6.00%	20.00%	16.00%	**58.00%**

Group 3	cotton_sheet_An	poly_sheet_An	boxing_An
cotton_sheet_An	**52.00%**	34.00%	14.00%
poly_sheet_An	12.00%	**72.00%**	16.00%
boxing_An	4.00%	40.00%	**56.00%**

Simulated Sounds:

Group 1	ball_bouncing	clapping	hand_rubbing	apple_slicing	jelly
ball_bouncing	**73.00%**	8.00%	7.00%	9.00%	3.00%
clapping	4.00%	**70.67%**	14.67%	5.33%	5.33%
hand_rubbing	1.33%	6.67%	**66.67%**	21.33%	4.00%
apple_slicing	2.67%	5.33%	16.00%	**74.67%**	1.33%
jelly	4.00%	5.33%	9.33%	9.33%	**72.00%**

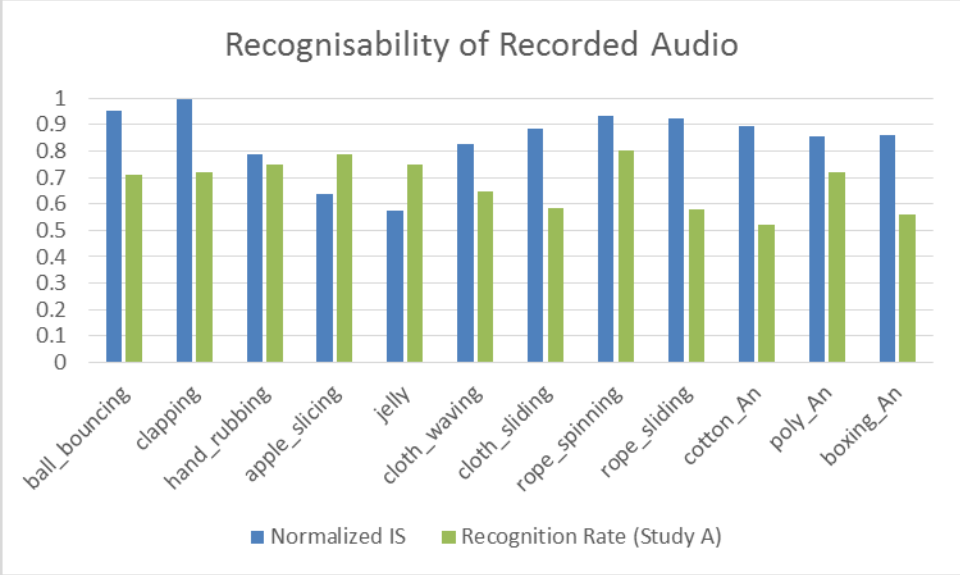
Group 2	cloth_waving	rope_spinning	cloth_sliding	rope_sliding
cloth_waving	**64.00%**	8.80%	16.80%	10.40%
rope_spinning	4.00%	**58.00%**	10.00%	28.00%
cloth_sliding	16.80%	13.60%	**57.60%**	12.00%
rope_sliding	10.00%	16.00%	20.00%	**54.00%**

Group 3	cotton_sheet_An	poly_sheet_An	boxing_An
cotton_sheet_An	**48.00%**	30.00%	22.00%
poly_sheet_An	8.00%	**70.00%**	22.00%
boxing_An	8.00%	28.00%	**64.00%**

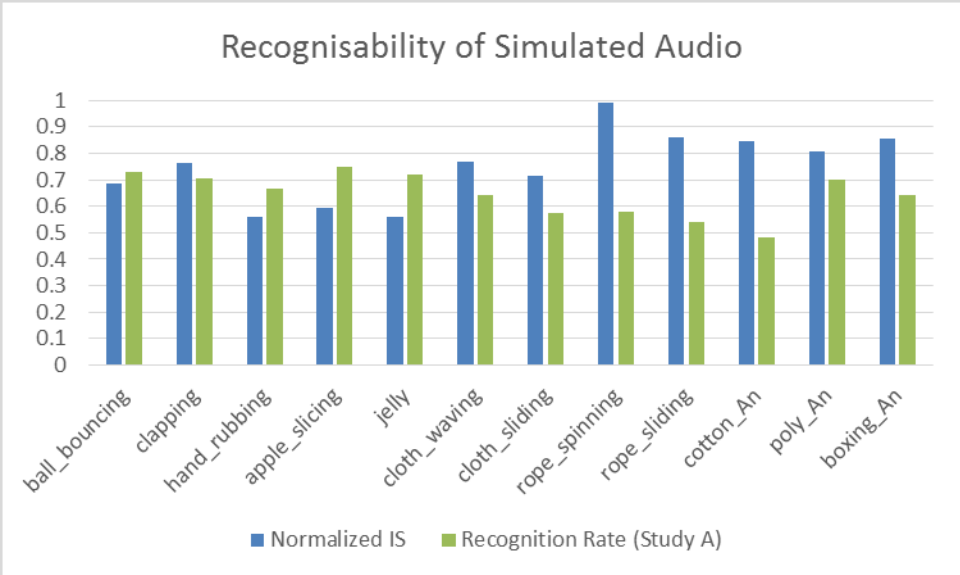
C.2 Data Normalization and Comparison

Recognisability: Normalized IS (0-1) vs. Recognition Rate (Study A)

Recorded Sounds:

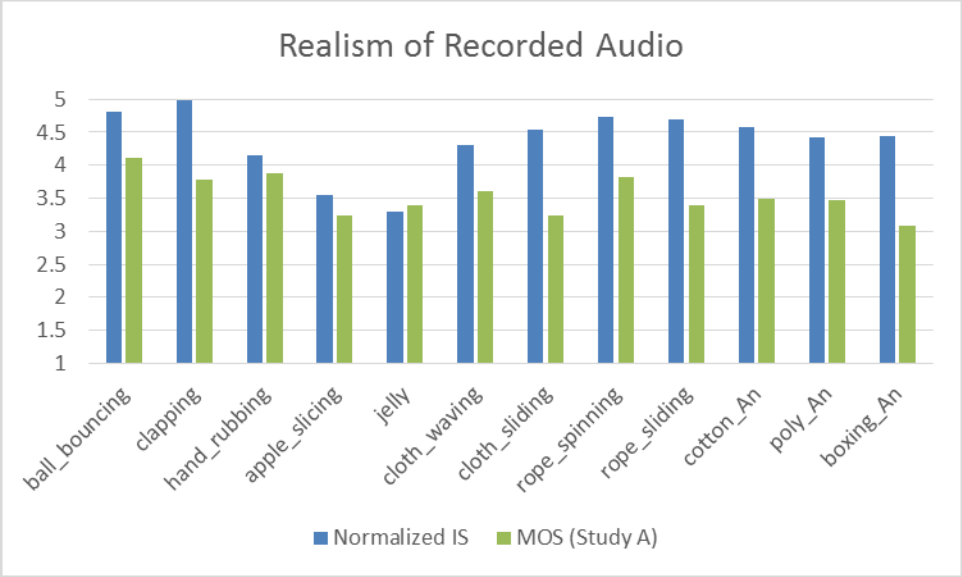


Simulated Sounds:

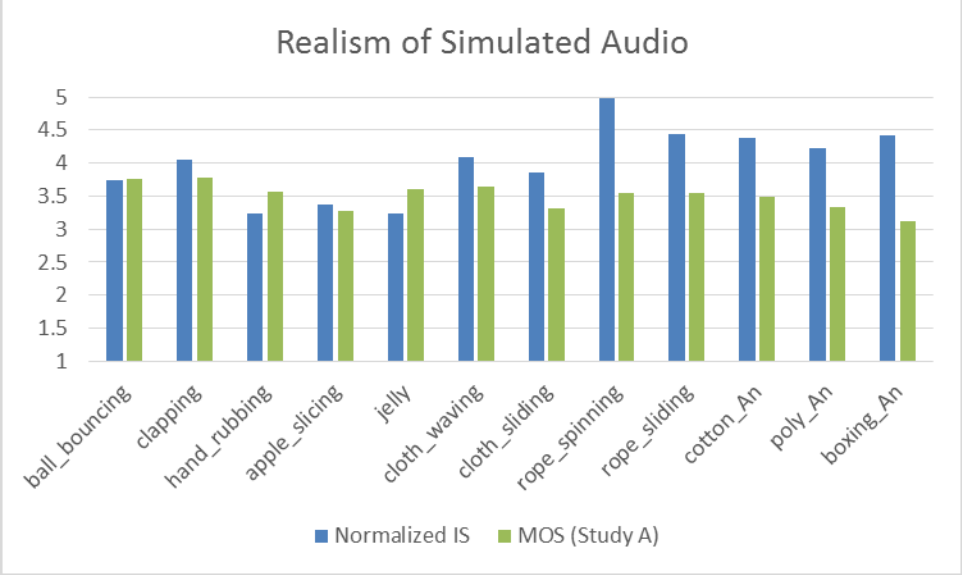


Realism: Normalized IS (1-5) vs. MOS (Study A)

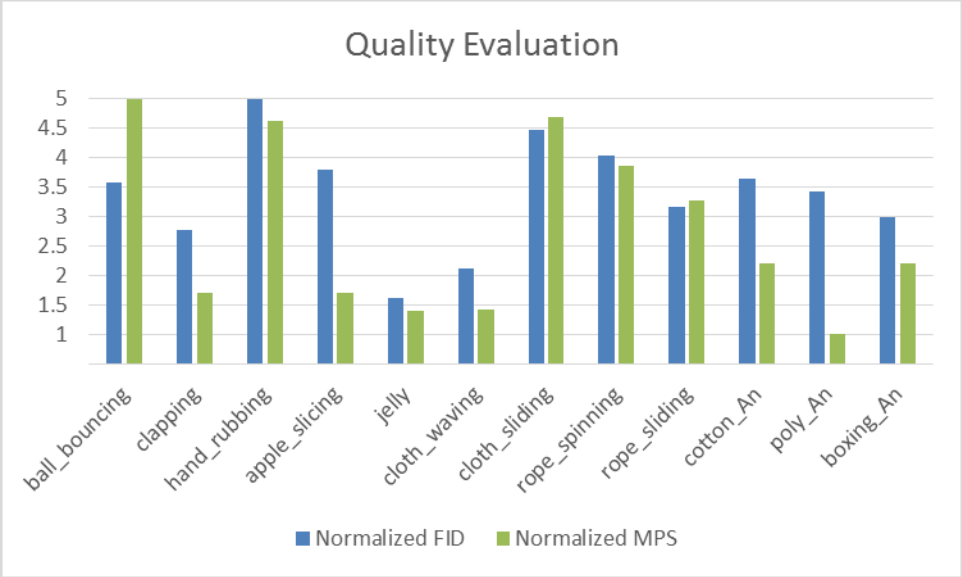
Recorded Sounds:



Simulated Sounds:



Quality: Normalized FID (1-5) vs. Normalized MPS (Study B)



References

[1] A. Farnell, *Designing Sound*. The MIT Press, 2010.

[2] B. Cowan, D. Rojas, B. Kapralos, F. Moussa, and A. Dubrowski, "Effects of sound on visual realism perception and task performance," *Vis. Comput.*, vol. 31, no. 9, pp. 1207–1216, 2015.

[3] T. Takala and J. Hahn, "Sound rendering," *ACM SIGGRAPH Comput. Graph.*, vol. 26, pp. 211–220, 1992.

[4] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 9, no. 1, pp. 1–22, 2013.

[5] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Pearson, 2010.

[6] P. R. Cook, *Real Sound Synthesis for Interactive Applications*. A K Peters, 2002.

[7] A. Farnell, "An introduction to procedural audio and its application in computer games," *Audio Most. Conf.*, no. September, pp. 1–31, 2007.

[8] M. Taylor, "RESound : Interactive Sound Rendering for Dynamic Virtual Environments," *17th Int. ACM Conf. Multimed. 2009*, pp. 271–280, 2009.

[9] F. Trebien and M. M. Oliveira, "Realistic real-time sound re-synthesis and processing for interactive virtual worlds," *Vis. Comput.*, vol. 25, no. 5–7, pp. 469–477, 2009.

[10] L. Pruvost, B. Scherrer, M. Aramaki, S. Ystad, and R. Kronland-Martinet, "Perception-based interactive sound synthesis of morphing solids' interactions," *SIGGRAPH Asia 2015 Tech. Briefs*, pp. 1–4, 2015.

[11] S. F. F. Gibson and B. Mirtich, "A Survey of Deformable Modeling in Computer Graphics," *Merl - a Mitsubishi Electr. Res. Lab.*, pp. 1–31, 1997.

[12] A. Nealen, M. Muller, R. Keiser, E. Boxerman, and M. Carlson, "Physically Based Deformable Models in Computer Graphics," *Comput. Graph. Forum*, vol. 25, no. 4, pp. 809–836, 2006.

[13] M. MacKlin, M. Muller, N. Chentanez, and T. Y. Kim, "Unified particle physics for real-time applications," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.

[14] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 205–214, 1987.

[15] D. Terzopoulos and K. Fleischer, "Modeling inelastic deformation: viscoelasticity, plasticity, fracture," *Comput. Graph.*, vol. 22, no. 4, pp. 269–278, 1988.

[16] F. Su and C. Joslin, "Procedurally-Generated Audio for Soft-Body Animations," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, 2018, pp. 1–8.

[17] F. Su and C. Joslin, "Toward Generating Realistic Sounds for Soft Bodies : A Review," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, 2019, pp. 199–206.

[18] F. Su and C. Joslin, "Procedural Sound Generation for Soft Bodies in Video Games," in *MIG '19: Motion, Interaction and Games*, 2019, pp. 1–12.

[19] K. van den Doel and D. K. Pai, “Synthesis of shape dependent sounds with physical modeling,” *Proc. Int. Conf. Audit. Disp.*, 1996.

[20] K. van den Doel, P. G. Kry, and D. K. Pai, “FoleyAutomatic,” *Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH ’01*, pp. 537–544, 2001.

[21] K. Van Den Doel and D. Pai, “Modal synthesis for vibrating objects,” *Audio Anectodes. AK Peter, Natick, MA*, pp. 1–8, 2003.

[22] J. F. O’Brien, P. R. Cook, and G. Essl, “Synthesizing sounds from physically based motion,” *Comput. Graph. Interact. Tech.*, pp. 529–536, 2001.

[23] J. F. O’Brien, C. Shen, and C. M. Gatchalian, “Synthesizing sounds from rigid-body simulations,” *Proc. 2002 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.*, p. 175, 2002.

[24] D. James, J. Barbic, and D. Pai, “Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources,” *ACM Trans. Graph.*, vol. 1, no. 212, p. 9, 2006.

[25] D. Li, Y. Fei, and C. Zheng, “Interactive Acoustic Transfer Approximation for Modal Sound,” *ACM Trans. Graph.*, vol. 35, no. 1, pp. 1–16, 2015.

[26] J. Wang and D. L. James, “KleinPAT: Optimal Mode Conflation For Time-Domain Precomputation Of Acoustic Transfer,” *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.

[27] J. K. Hahn, H. Fouad, L. Gritz, and J. W. Lee, “Integrating Sounds and Motions in Virtual Environments,” *Presence Teleoperators Virtual Environ.*, vol. 7, no. 1, pp. 67–77, 1998.

[28] N. Raghuvanshi and M. Lin, “Interactive sound synthesis for large scale environments,” *Proc. 2006 Symp. Interact. 3D Graph. games*, vol. 1, no. March, pp. 14–17, 2006.

[29] Z. Ren, H. Yeh, and M. C. Lin, “Synthesizing contact sounds between textured models,” *Proc. - IEEE Virtual Real.*, pp. 139–146, 2010.

[30] N. Tsingos, E. Gallo, and G. Drettakis, “Perceptual audio rendering of complex virtual environments,” *ACM Trans. Graph.*, vol. 23, no. 3, p. 249, 2004.

[31] N. Bonneel, G. Drettakis, N. Tsingos, I. Viaud-Delmon, and D. James, “Fast modal sounds with scalable frequency-domain synthesis,” *ACM Trans. Graph.*, vol. 27, no. 3, p. 1, 2008.

[32] T. R. Langlois, S. S. An, K. K. Jin, and D. L. James, “Eigenmode compression for modal sound models,” *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–9, 2014.

[33] C. Zheng and D. L. James, “Rigid-body fracture sound with precomputed soundbanks,” *ACM Trans. Graph.*, vol. 29, no. 4, p. 1, 2010.

[34] J. N. Chadwick, S. S. An, and D. L. James, “Harmonic Shells: A Practical Nonlinear Sound Model for Near-Rigid Thin Shells,” *ACM SIGGRAPH Asia 2009 Pap. - SIGGRAPH Asia ’09*, vol. 28, no. 5, p. 1, 2009.

[35] G. Cirio, D. Li, E. Grinsprun, M. A. Otaduy, and C. Zheng, “Crumpling Sound Synthesis,” *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–11, 2016.

[36] G. Cirio, A. Qu, G. Drettakis, E. Grinspun, and C. Zheng, “Multi-Scale Simulation of Nonlinear Thin-Shell Sound with Wave Turbulence,” *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–14, 2018.

[37] C. Zheng and D. L. James, “Toward high-quality modal contact sound,” *ACM Trans. Graph.*, vol. 30, no. 4, p. 1, 2011.

[38] J. N. Chadwick, C. Zheng, and D. L. James, “Precomputed acceleration noise for improved rigid-body sound,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–9, 2012.

[39] J. N. Chadwick, C. Zheng, and D. L. James, “Faster acceleration noise for multibody animations using precomputed soundbanks,” *ACM/Eurographics Symp. Comput. Animat.*, pp. 265–273, 2012.

[40] J.-H. Wang, A. Qu, T. R. Langlois, and D. L. James, “Toward Wave-based Sound Synthesis for Computer Animation,” *ACM Trans. Graph. Artic.*, vol. 37, no. 109, 2018.

[41] K. Doel, “Physically Based Models for Liquid Sounds,” *ACM Trans. Appl. Percept.*, vol. 2, no. 4, pp. 534–546, 2005.

[42] C. Zheng and D. L. James, “Harmonic fluids,” *ACM Trans. Graph.*, vol. 28, no. 3, p. 1, 2009.

[43] W. Moss, H. Yeh, J.-M. Hong, M. C. Lin, and D. Manocha, “Sounding liquids,” *ACM Trans. Graph.*, vol. 29, no. 3, pp. 1–13, 2010.

[44] T. R. Langlois, C. Zheng, and D. L. James, “Toward animating water with complex acoustic bubbles,” *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–13, 2016.

[45] S. Liu, H. Cheng, and Y. Tong, “Physically-based statistical simulation of rain sound,” *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.

[46] Y. Dobashi, T. Yamamoto, and T. Nishita, “Real-time rendering of aerodynamic sound using sound textures based on computational fluid dynamics,” *ACM Trans. Graph.*, vol. 22, p. 732, 2003.

[47] Y. Dobashi, T. Yamamoto, and T. Nishita, “Synthesizing sound from turbulent field using sound textures for interactive fluid simulation,” *Comput. Graph. Forum*, vol. 23, no. 3 SPEC. ISS., pp. 539–545, 2004.

[48] E. Schweickart, D. L. James, and S. Marschner, “Animating Elastic Rods with Sound,” *ACM Trans. Graph.*, vol. 36, no. 4, p. 115, 2017.

[49] M. Cardle, S. Brooks, Z. Bar-Joseph, and P. Robinson, “Sound-by-numbers: motion-driven sound synthesis,” *Proc. 2003 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.*, pp. 349–356, 2003.

[50] C. Roads, *Microsound*. The MIT Press, 2004.

[51] C. Picard, N. Tsingos, and F. Faure, “Retargetting example sounds to interactive physics-driven animations,” *Audio Eng. Soc. Conf.*, pp. 1–8, 2009.

[52] S. S. An, D. L. James, and S. Marschner, “Motion-driven concatenative synthesis of cloth sounds,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, 2012.

[53] D. Schwarz, “A System for Data-Driven Concatenative Sound Synthesis,” *Digit. Audio Eff.*, pp. 97–102, 2000.

[54] C. Schreck, D. Rohmer, D. James, S. Hahmann, and M.-P. Cani, “Real-time sound synthesis for paper material based on geometric analysis,” *Eurographics/ACM SIGGRAPH Symp. Comput. Animat.*, 2016.

[55] J. N. Chadwick and D. L. James, “Animating fire with sound,” *ACM Trans. Graph.*, vol. 30, no. 4, p. 1, 2011.

[56] M. Aramaki and R. Kronland-Martinet, “Analysis-synthesis of impact sounds,” *Eur. Signal Process. Conf.*, vol. 06-10-Sept, pp. 1769–1772, 2015.

[57] J. L. Richmond and D. K. Pai, “Active measurement of contact sounds,” *IEEE Int. Conf. Robot. Autom.*, vol. 3, no. April, pp. 2146–2152, 2000.

[58] D. Pai, K. Doel, D. James, and J. Lang, “Scanning Physical Interaction Behavior of 3D Objects,” *Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 87–96, 2001.

[59] D. B. Lloyd, N. Raghuvanshi, and N. K. Govindaraju, “Sound synthesis for impact sounds in video games,” *Proc. Symp. Interact. 3D Graph. Games*, pp. 55–62, 2011.

[60] Z. Ren, H. Yeh, and M. C. Lin, “Example-guided physically based modal sound synthesis,” *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–16, 2013.

[61] K. van den Doel, D. K. Pai, T. Adam, L. Kortchmar, and K. Pichora-Fuller, “Measurements of perceptual quality of contact sound models,” in *Proc. of the International Conference on Auditory Display (ICAD 2002)*, 2002, pp. 345–349.

[62] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Y. Chang, and T. Sainath, “Deep Learning for Audio Signal Processing,” *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 2, pp. 206–219, 2019.

[63] V. Chandrasekhar, M. Sharifi, and D. A. Ross, “Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications,” *Proc. 12th Int. Soc. Music Inf. Retr. Conf. ISMIR 2011*, pp. 801–806, 2011.

[64] A. L.-C. Wang, “An Industrial-Strength Audio Search Algorithm,” *Ismir*, vol. 2003, pp. 7–13, 2003.

[65] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, “Robust Sound Event Classification Using Deep Neural Networks,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 23, no. 3, pp. 540–552, 2015.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. London: The MIT Press, 2016.

[67] H. Lee, L. Yan, P. Pham, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” *Adv. Neural Inf. Process. Syst. 22 - Proc. 2009 Conf.*, pp. 1096–1104, 2009.

[68] I. J. Goodfellow, J. Pouget-abadie, M. Mirza, B. Xu, and D. Warde-farley, “Generative Adversarial Nets,” *Adv. Neural Inf. Process. Syst.*, pp. 2672–2680, 2014.

[69] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” *Adv. Neural Inf. Process. Syst.*, pp. 2234–2242, 2016.

[70] C. Zheng, “Physics-Based Sound Rendering for Computer Animation,” Cornell University, 2012.

[71] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[72] O. Abdel-Hamid, A. R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, 2014.

[73] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” *IEEE Int. Work. Mach. Learn. Signal Process. MLSP*, vol. 2015-Novem, pp. 1–6, 2015.

[74] V. Boddapati, A. Petef, J. Rasmusson, and L. Lundberg, “Classifying environmental sounds using image recognition networks,” *Procedia Comput. Sci.*, vol. 112, pp. 2048–2056, 2017.

[75] S. Hershey *et al.*, “CNN architectures for large-scale audio classification,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 131–135, 2017.

[76] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, “Large-scale video classification with convolutional neural networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1725–1732, 2014.

[77] O. Gencoglu, T. Virtanen, and H. Huttunen, “Recogntion of acoustic events using deep neural networks,” *2014 22nd Eur. Signal Process. Conf.*, pp. 506–510, 2014.

[78] H. Zhang, I. McLoughlin, and Y. Song, “Robust sound event recognition using convolutional neural networks,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2015-Augus, pp. 559–563, 2015.

[79] N. Takahashi, M. Gygli, B. Pfister, and G. Luc Van, “Deep convolutional neural networks and data augmentation for acoustic event detection,” in *Interspeech 2016*, 2016, pp. 2982–2986.

[80] L. Hertel, H. Phan, and A. Mertins, “Comparing time and frequency domain for audio event recognition using deep learning,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2016-Octob, pp. 3407–3411, 2016.

[81] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2016-May, pp. 6440–6444, 2016.

[82] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” *2009 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 248–255, 2010.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[84] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016.

[85] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” *7th Int. Conf. Learn. Represent. ICLR 2019*, pp. 1–16, 2019.

[86] L. Theis, A. Van Den Oord, and M. Bethge, “A note on the evaluation of generative models,” *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*, pp. 1–10, 2016.

[87] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 6627–6638, 2017.

[88] L. Peltola, C. Erkut, P. R. Cook, and V. Välimaki, “Synthesis of hand clapping sounds,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 15, no. 3, pp. 1021–1029, 2007.

[89] B. H. Repp, “The sound of two hands clapping: An exploratory study,” *J. Acoust. Soc. Am.*, vol. 81, no. 4, pp. 1100–1109, 1987.

[90] C. Clapham and J. Nicholson, *The Concise Oxford Dictionary of Mathematics*, 5th ed. Oxford University Press, 2014.

[91] T. Igarashi and J. Mitani, “Apparent layer operations for the manipulation of deformable objects,” *ACM SIGGRAPH 2010 Pap. SIGGRAPH 2010*, vol. 1, no. 212, pp. 1–7, 2010.

[92] R. B. Fisher, *From surfaces to objects: computer vision and three dimensional scene analysis*. New York: Wiley, 1989.

[93] M. H. Davis, A. Khotanzad, D. P. Flamig, and S. E. Harms, “Curvature Measurement of 3D Objects: Evaluation and Comparison of Three Methods,” *Electr. Eng.*, no. 1, pp. 627–630, 1995.

[94] C. P. Brown and R. O. Duda, “An efficient HRTF model for 3-D sound,” in *Proceedings of 1997 Workshop on Applications of Signal Processing to Audio and Acoustics*, 1997, pp. 4-pp.

[95] C. P. van den Doel, “Sound synthesis for virtual reality and computer games,” University of British Columbia, 1998.

[96] C. Szegedy *et al.*, “Going Deeper with Convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015.

[97] J. F. Gemmeke *et al.*, “Audio Set: An ontology and human-labeled dataset for audio events,” *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 776–780, 2017.

[98] J. Salamon, C. Jacoby, and J. P. Bello, “A Dataset and Taxonomy for Urban Sound Research,” *Proc. 2014 ACM Conf. Multimed.*, no. 1, pp. 1041–1044, 2014.

[99] I. The MathWorks, “Deep Learning Toolbox.” Natick, Massachusetts, United State, 2020.

[100] T. R. Langlois and D. L. James, “Inverse-Foley animation,” *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–11, 2014.

[101] S. Barratt and R. Sharma, “A Note on the Inception Score,” *Proc. ICML 2018 Work. Theor. Found. Appl. Deep Gener. Model.*, 2018.

[102] B. Cowan and B. Kapralos, “Spatial sound rendering for dynamic virtual environments,” *2013 18th Int. Conf. Digit. Signal Process. DSP 2013*, pp. 1–6, 2013.

[103] H. Jo, M. Imran, and J. Y. Jeon, “Methodology for virtual sound synthesis for graphical object interactions,” *24th Int. Congr. Sound Vib. ICSV 2017*, 2017.

[104] C. Zheng and D. L. James, “Energy-based self-collision culling for arbitrary mesh deformations,” *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–12, 2012.

[105] G. Strobl, G. Eckel, and D. Rocchesso, “Sound Texture Modeling : a Survey,” *Proc. 2006 Sound Music Comput. Int. Conf.*, pp. 3–7, 2006.

[106] N. Saint-arnaud and K. Popat, “Analysis and Synthesis of Sound Textures,” *Readings Comput. Audit. Scene Anal.*, pp. 125--131, 1995.

[107] Z. Bar-Joseph, D. Lischinski, M. Werman, S. Dubnov, and R. ELYANIV, “Granular synthesis of sound textures using statistical learning,” *Proceedings of the International Computer Music Conference*. pp. 178–181, 1999.

[108] S. Dubnov, Z. Bar-joseph, R. El-yaniv, D. Lischinski, and M. Werman, "Synthesizing Sound Textures through Wavelet Tree Learning," *IEEE Comput. Graph. Appl.*, vol. 22, no. August, pp. 38–48, 2002.