DiffSound: Differentiable Modal Sound Rendering and Inverse Rendering for Diverse Inference Tasks

Xutong Jin*
jinxutong@pku.edu.cn
School of Computer Science, Peking University
China

Chenxi Xu*
2301213239@pku.edu.cn
School of Computer Science, Peking University
China

Ruohan Gao
rhgao@umd.edu
University of Maryland, College Park
U.S.A

Jiajun Wu
jiajunwu@cs.stanford.edu
Stanford University
U.S.A

Guoping Wang
wgp@pku.edu.cn
School of Computer Science, Peking University
China

Sheng Li†
lisheng@pku.edu.cn
School of Computer Science, Peking University
China

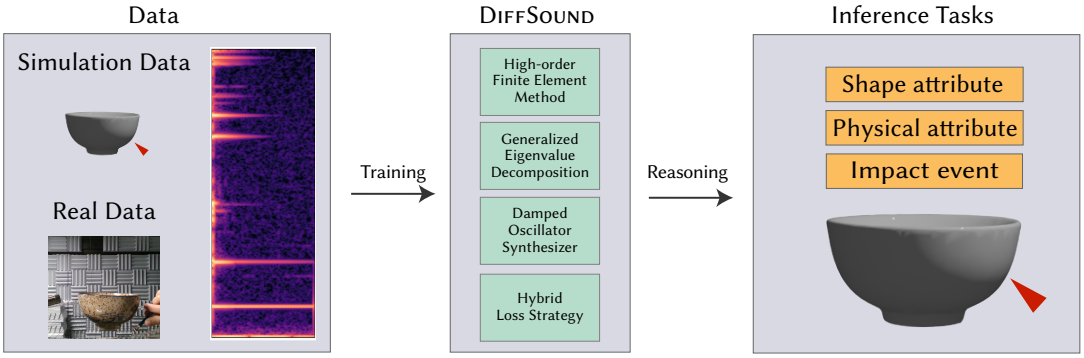

Figure 1: We introduce DiffSound, a differentiable sound rendering framework for physics-based modal sound synthesis. It can infer a variety of physical and shape attributes, such as material, volumetric thickness, geometric shape, and impact positions of the object, from both simulated datasets and real sound recordings, enabling a series of inverse rendering applications.

ABSTRACT

Accurately estimating and simulating the physical properties of objects from real-world sound recordings is of great practical importance in the fields of vision, graphics, and robotics. However, the progress in these directions has been limited—prior differentiable rigid or soft body simulation techniques cannot be directly applied to modal sound synthesis due to the high sampling rate of audio, while previous audio synthesizers often do not fully model the accurate physical properties of the sounding objects. We propose DiffSound, a differentiable sound rendering framework for physics-based modal sound synthesis, which is based on an implicit shape representation, a new high-order finite element analysis module, and a differentiable audio synthesizer. Our framework can solve a wide range of inverse problems thanks to the differentiability of the entire pipeline, including physical parameter estimation, geometric shape reasoning, and impact position prediction. Experimental results demonstrate the effectiveness of our approach, highlighting its ability to accurately reproduce the target sound in a physics-based manner. DiffSound serves as a valuable tool for various sound synthesis and analysis applications.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**.

KEYWORDS

sound synthesis, differentiable simulation, modal analysis, vibration, audio

ACM Reference Format:
Xutong Jin, Chenxi Xu, Ruohan Gao, Jiajun Wu, Guoping Wang, and Sheng Li. 2024. DiffSound: Differentiable Modal Sound Rendering and Inverse Rendering for Diverse Inference Tasks. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3641519.3657493

*Xutong Jin and Chenxi Xu have equal contributions.
†Sheng Li is the corresponding author.

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657493

1 INTRODUCTION

The concept of differentiable simulation has become increasingly popular in the graphics and machine learning communities in recent years [de Avila Belbute-Peres et al. 2018; Degrave et al. 2019; Popović et al. 2003; Qiao et al. 2020; Toussaint et al. 2019; Xu et al. 2021]. A differentiable simulation framework has the benefit of allowing for gradient-based optimization and thus can be easily integrated into a neural network for end-to-end learning.

Our work focuses on differentiable sound rendering, which addresses a unique challenge compared to standard differentiable rigid or soft body simulations [Degrave et al. 2019; Du et al. 2021; Geilinger et al. 2020; Hu et al. 2020; Qiao et al. 2020; Xu et al. 2021] due to the high sampling rate of sound. While previous audio synthesizers [Clarke et al. 2021; Engel et al. 2020] can optimize for many audio and physical-based properties, they are unable to explicitly model more fundamental physical properties such as Young's modulus, Poisson's ratio, size, or shape of the object, and the impact position, which are all critical for realistic modal sound synthesis.

Inferring these objects' properties from real sound recordings can potentially enable various Real-to-Sim applications. For example, we can accurately infer material parameters from real-world recordings and use them to re-create realistic virtual objects, such as those in Clarke et al. [2023]; Gao et al. [2021, 2023, 2022]. We can also leverage a differentiable sound rendering framework to design the shape and material of virtual objects to produce the desired sound, and then transfer the results back to real objects using 3D printing technology [Bharaj et al. 2015]. The information about an object's shape, material, and impact position can also complement visual perception, particularly in cases of low visual resolution or poor lighting, especially for multisensory robotic applications [Clarke et al. 2021; Li et al. 2022b].

Towards this end, we introduce DiffSound, a differentiable simulation framework for modal sound synthesis that employs a high-order finite element method for physics-based modeling. It not only is fully-differentiable and allows for efficient end-to-end optimization, but also establishes a seamless connection between the recorded audio and the fundamental physical properties of real-world objects. Our DiffSound consists of three main components. First, we propose a hybrid shape representation that combines implicit neural representation and explicit 3D tetrahedral mesh representation. Second, we introduce a high-order finite element analysis module that allows for incorporating different material and shape parameters. Finally, we design a differentiable audio synthesizer with a hybrid loss strategy to enable smooth optimization of the entire differentiable simulation pipeline.

We demonstrate the effectiveness of our method through a wide range of inverse rendering tasks. Our differentiable framework can accurately estimate the attributes of the sounding objects, such as the material parameters, identify the impact positions and amplitudes of the physical interactions, and infer the object shape characteristics, including volumetric thickness, and geometric form, on both synthetic and real-world audio datasets. Notably, while limited prior work has proposed to infer material properties from sound [Ren et al. 2013], to our best knowledge, our work marks the first attempt to estimate an object's thickness, precise geometric shape and impact position purely through sound analysis.

2 RELATED WORK

Modal Sound Synthesis. Modal sound synthesis is a technique that has been used to synthesize sounds of rigid bodies [O'Brien et al. 2002; Raghuvanshi and Lin 2006; van den Doel et al. 2001]. These methods compute the vibration modes of a 3D object through a generalized eigenvalue decomposition. Based on the basic modal sound method, many complex sound phenomena can be simulated, such as knocking, sliding, and friction sound [van den Doel et al. 2001], acceleration noise [Chadwick et al. 2012], complex damping sound [Sterling et al. 2019], and high-quality contact sound [Zheng and James 2011].

Related to prior studies on estimating material parameters from pre-recorded audio [Ren et al. 2013; Zhang et al. 2017], our work differs by providing an end-to-end optimization-based solution, enhancing accuracy. Unlike previous methods optimizing object shape for desired sound [Bharaj et al. 2015], our approach optimizes all sound modes, not just the fundamental frequency. Additionally, it offers greater flexibility in shape optimization, surpassing simple scaling and stretching.

High-Order FEM. In engineering, higher-order methods are often preferred over lower-order methods due to their superior accuracy and convergence properties. In computer graphics, finite element methods (FEM) with linear shape functions are prevalent due to their simplicity and computational efficiency. While limited prior work demonstrates that higher-order methods have the potential to produce better simulation results [Bargteil and Cohen 2014; Longva et al. 2020; Mezger et al. 2008; Schneider et al. 2019], they are not commonly used in the field.

To the best of our knowledge, the sole previous attempt [Bharaj et al. 2015] that incorporates high-order FEM in modal sound synthesis directly employs the engineering software COMSOL [COMSOL AB, Stockholm, Sweden 2005] to obtain the results. In contrast, within our differentiable framework, we newly implement a high-order FEM module to guarantee both high-quality sound rendering and differentiability.

Differentiable Simulation. Differentiable simulation has recently gained much popularity in the graphics and machine learning communities. Several advances have been made in this field with differentiable simulators designed for rigid-body dynamics [Cleac'h et al. 2023; de Avila Belbute-Peres et al. 2018; Degrave et al. 2019; Popović et al. 2003; Qiao et al. 2020; Toussaint et al. 2019; Xu et al. 2021], soft-body dynamics [Du et al. 2021; Geilinger et al. 2020; Hahn et al. 2019; Hu et al. 2020, 2019], fluid dynamics [Holl et al. 2020; McNamara et al. 2004; Schenck and Fox 2018; Treuille et al. 2003; Wojtan et al. 2006], and cloth [Li et al. 2022a; Liang et al. 2019; Murthy et al. 2021].

There are also differentiable rendering methods proposed for signal processing [Engel et al. 2020] and modeling impact sound [Clarke et al. 2021]. These methods can capture various physics-based properties, such as modal response and force profiles. However, they do not explicitly consider the fundamental physical properties of objects, such as shape, material, and impact position. Another promising approach uses neural networks to approximate the modal analysis process [Jin et al. 2020, 2022]. Although neural networks are

inherently differentiable, ensuring physical accuracy can be challenging, and accurate modal analysis cannot be trivially achieved just through neural network optimization.

3 DIFFERENTIABLE MODAL SOUND RENDERING

First, we give an overview of our full model (Sec. 3.1). Then, we describe the differentiable tetrahedral mesh representation (Sec. 3.2), differentiable high-order finite element method (FEM) for modal analysis (Sec. 3.3), and hybrid loss strategy for optimizing all learnable modules (Sec. 3.4).

3.1 Method Overview

Differentiable methods in machine learning, like our sound synthesis method DiffSound, allow for computing output gradients from model parameters. This facilitates parameter optimization using gradient-based algorithms, with our method ensuring a differentiable pipeline from 3D mesh input to modal sound output.

Specifically, assuming θ represents the learnable parameters in our framework. For an input mesh m, we introduce a differentiable explicit tetrahedral mesh generator G_{θ_G}, which transforms the input mesh into an explicit tetrahedral mesh $m_{tet} = G_{\theta_G}(m)$. The generated tetrahedral mesh is constrained by generator parameters θ_G, such as geometric shapes and thickness.

Next, we introduce a differentiable high-order FEM module, including a differentiable FEM matrix assembler A_{θ_A} and a differentiable generalized eigenvalue decomposer D. The matrix assembler takes an explicit tetrahedral mesh as input, and outputs its mass matrix and stiffness matrix $\mathbf{M}, \mathbf{K} = A_{\theta_A}(m_{tet})$ constrained by parameters θ_A, including Young's modulus and Poisson's ratio. The generalized eigenvalue decomposer D takes $\mathbf{M}, \mathbf{K}$ as input and outputs its eigenvalues corresponding to $\mathbf{KU} = \mathbf{MU\Lambda}$, denoted as $\lambda = diag(\mathbf{\Lambda}) = D(\mathbf{M}, \mathbf{K})$.

Building upon this, we introduce a differentiable additive synthesizer S_{θ_S}, which takes eigenvalues λ as input and synthesizes its modal sound $a_{syn} = S_{\theta_S}(\lambda)$ with the constraint of θ_S, such as damping coefficients and mode amplitudes. Finally, we introduce a hybrid loss function L that compares the synthesized audio a_{syn} generated from the aforementioned process with the ground-truth audio a_{gt}, resulting $loss = L(a_{syn}, a_{gt})$. For simplicity, we consolidate the process of differentiable sound synthesis by the mesh m into a function F: $a_{syn} = F_\theta(m) = S_{\theta_S}(D(A_{\theta_A}(G_{\theta_G}(m))))$. At this point, the optimization target is as follows:

$$\theta^* = \text{argmin}_\theta(L(F_\theta(m), a_{gt})). \tag{1}$$

In certain tasks, it is often required to optimize the parameters of just one module while the parameters of other modules remain fixed. The schematic representation of our approach's methodology is depicted in Figure 2.

3.2 Differentiable Tetrahedral Representation

We propose a differentiable tetrahedral mesh representation tailored for our differentiable sound rendering, building upon the foundation of Deep Marching Tetrahedra (DMTet) [Munkberg et al. 2022; Shen et al. 2021]. Our approach involves the representation of a shape through a Signed Distance Field (SDF) implicitly encoded by a Multilayer Perceptron (MLP) (Sec. 3.2.1), which is then transformed into an explicit tetrahedral mesh using a deformable tetrahedral grid (Sec. 3.2.2).

3.2.1 Implicit Neural Representation. Given the inherent challenge of precisely associating the sound of an object with its exact shape, there can be significant ambiguity in the resulting geometry when optimizing purely from sound. To address this, we utilize a Multilayer Perceptron (MLP) to parameterize the SDF values. This implicit parameterization effectively serves to regulate both the SDF and the overall smoothness of the reconstructed shape. Additionally, the degree of smoothness can be fine-tuned by varying the frequency of the positional encoding, following Neural Radiance Fields [Mildenhall et al. 2020], applied to the inputs of the MLP.

3.2.2 Implicit to Explicit Representation. We adopt the Marching Tetrahedra (MT) algorithm [Doi and Koide 1991] to transform encoded Signed Distance Function (SDF) data into explicit tetrahedral meshes. Our approach allows background tetrahedral cell vertices to deform within half-cell size limits, enhancing geometric expression. Using SDF values from the MLP for vertices in a tetrahedron, MT discerns surface topology based on SDF sign variations. Our method focuses on identifying internal tetrahedra rather than surface topology, resulting in five configurations due to rotational symmetry (see Figure 3). Surface vertex locations are determined by linear interpolation along tetrahedron edges, akin to DMTet's approach [Munkberg et al. 2022; Shen et al. 2021]. For complex internal sub-regions, we further subdivide into smaller tetrahedrons. To minimize high-frequency noise impact in sound optimization, we extract the largest connected tetrahedral mesh, discarding small fragments. Despite the potential sudden appearance or disappearance of fragments, their existence is consistently determined during the computation of the gradient in each step. Consequently, our approach ensures the accurate calculation of the gradient.

3.3 Differentiable High-order FEM

Prior studies [Bharaj et al. 2015; Hughes 2012] have noted the limitations of linear tetrahedral finite elements in producing accurate solutions, even with refined simulation discretization. In this work, we propose the use of differentiable high-order FEM for greater accuracy and generality.

We compute the mass and stiffness matrices for the tetrahedral mesh (introduced in Sec. 3.2 above), which are made differentiable with respect to the material coefficients, namely Young's modulus, density, and Poisson's ratio as introduced in Sec. 3.3.1. Subsequently, in Sec. 3.3.2, we compute the gradient from the eigenvalues obtained through eigendecomposition with respect to these two matrices. For a comprehensive derivation of these matrices, please refer to [Sifakis and Barbic 2012; Zhu 2018].

3.3.1 Mass and Stiffness Matrix. To obtain the mass matrix, we initially compute the element matrix for each individual tetrahedral element, followed by the assembly process to construct the mass matrix for the entire tetrahedral mesh. Let V denote the volume occupied by a tetrahedral element, ρ represents its density, and the shape function value at position x with respect to node i is denoted

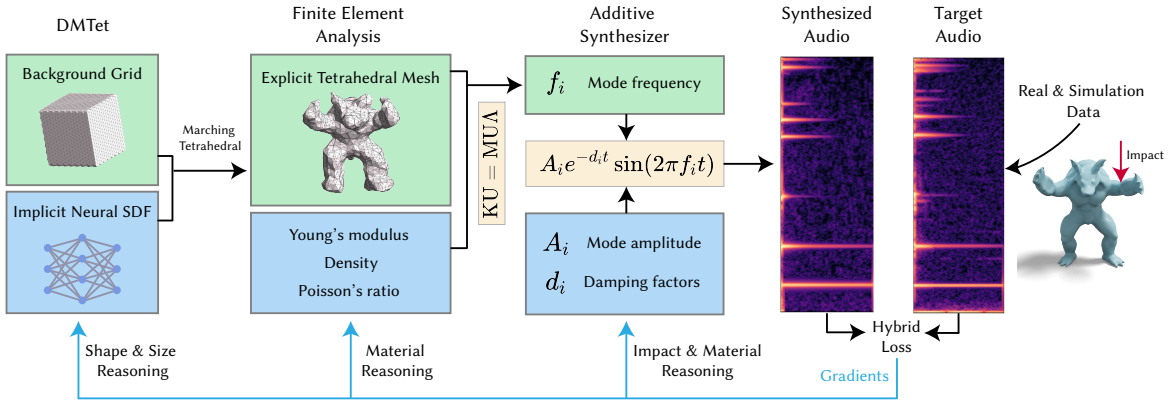

Figure 2: Our DiffSound differentiable simulation and inverse rendering pipeline. The differentiable tetrahedral mesh representation is employed to directly optimize the topology of a tetrahedral mesh. Subsequently, a differentiable high-order finite element analysis module is utilized to analyze the vibration frequencies of the tetrahedral mesh. Finally, a differentiable additive synthesizer is used to produce the impact sound with a hybrid loss function for optimizing all learnable modules. The learnable parameters, indicated by blue boxes, control module outputs in our differentiable framework. This enables gradient computation for hybrid loss, facilitating parameter optimization.

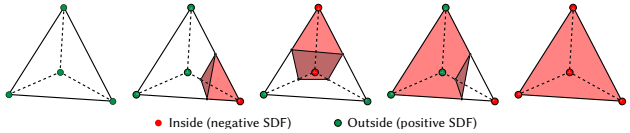

Figure 3: Five configurations of the interface between background tetrahedrons and internal ones. If the internal subregion is more complex than a tetrahedron, it should be subdivided into smaller tetrahedrons.

as $N_i(x)$. The element mass matrix $\mathbf{M}_e$ is defined as follows:

$$\mathbf{M}_e^{ij} = \rho \iiint_{x \in V} N_i(x) N_j(x) dx \, . \tag{2}$$

To compute this volume integral, we employ the Gaussian numerical integration method, selecting t Gaussian integration points g_k within the tetrahedral element, with corresponding Gaussian integration weights w_k. The unit mass matrix can be calculated as:

$$\mathbf{M}_e^{ij} = \rho V \sum_{k=1}^{t} N_i(g_k) N_j(g_k) w_k \, . \tag{3}$$

For a high-order tetrahedral element containing n nodes, the algorithm described above yields a unit mass matrix $\mathbf{M}_e$ of size $3n \times 3n$. Now, for the entire tetrahedral mesh with a total of m nodes, it is only necessary to add each element $\mathbf{M}_e^{ij}$ computed for each tetrahedron to the corresponding entries $\mathbf{M}^{ij}$ of the overall mesh's mass matrix $\mathbf{M}$. This assembles a $3m \times 3m$ mass matrix $\mathbf{M}$.

Following the defined process for the mass matrix, let E denote Young's modulus and ν denote Poisson's ratio. The element stiffness matrix $\mathbf{K}_e$ of size $3n \times 3n$ for a tetrahedral element is defined as:

$$\mathbf{K}_e = \sum_{k=0}^{t} w_k V \mathbf{D}(g_k)^T \mathbf{B}(E, \nu) \mathbf{D}(g_k) \, . \tag{4}$$

Here, $\mathbf{B}(E, \nu)$ is the elasticity matrix representing the material model, and we adopt the linear elastic model [Sifakis and Barbic 2012]. $\mathbf{D}(g_k)$ is a matrix derived from the shape functions at point g_k. To construct the overall stiffness matrix $\mathbf{K}$ for the entire tetrahedral mesh, we add each element in $\mathbf{K}_e$ computed for each tetrahedron to the corresponding entries of the overall mesh's stiffness matrix $\mathbf{K}$. This assembles a $3m \times 3m$ stiffness matrix $\mathbf{K}$.

We employ PyTorch [Paszke et al. 2017] to efficiently batch calculate both the element mass matrix and element stiffness matrix. Subsequently, these element matrices are assembled into global Coordinate Format (COO) sparse matrices for further processing. Notably, it is essential to highlight that these computations are automatically differentiable, enabled by PyTorch. Additionally, both the mass and stiffness matrices exhibit differentiability with respect to the material properties (ρ in the mass matrix and $\mathbf{B}(E, \nu)$ in the stiffness matrix), as well as the geometry derived from our differentiable tetrahedral mesh ($N_i(x)$ in the mass matrix and $\mathbf{D}(g_k)$ in the stiffness matrix, as well as V in both cases).

3.3.2 Eigenvalue Decomposition. Now, we perform a generalized eigenvalue decomposition on the mass and stiffness matrices as $\mathbf{KU} = \mathbf{MU\Lambda}$, where $\mathbf{U}$ is a stack of k eigenvectors, and $\mathbf{\Lambda}$ is the diagonal matrix of k eigenvalues. The i-th eigenvector, denoted as $\mathbf{u}_i$, represents the surface vibration distribution of the i-th mode, while the i-th eigenvalue, λ_i, determines its frequency and satisfies $\mathbf{K}\mathbf{u}_i = \lambda_i \mathbf{M}\mathbf{u}_i$. Since there is currently no eigenvalue decomposer that supports automatic differentiation, we thus derive the gradient relationship between eigenvalues and the mass and stiffness matrices as follows. Taking the derivative of both sides with respect to λ_i in the equation $\mathbf{K}\mathbf{u}_i = \lambda_i \mathbf{M}\mathbf{u}_i$, we obtain:

$$\partial \mathbf{K}\mathbf{u}_i + \mathbf{K}\partial \mathbf{u}_i = \lambda_i \mathbf{M}\partial \mathbf{u}_i + \lambda_i \partial \mathbf{M}\mathbf{u}_i + \partial \lambda_i \mathbf{M}\mathbf{u}_i \, , \tag{5}$$

By pre-multiplying both sides by $\mathbf{u}_i^T$ and rearranging the terms, we obtain:

$$\mathbf{u}_i^T (\partial \mathbf{K} - \lambda_i \partial \mathbf{M}) \mathbf{u}_i + \mathbf{u}_i^T (\mathbf{K} - \lambda_i \mathbf{M}) \partial \mathbf{u}_i = \mathbf{u}_i^T \partial \lambda_i \mathbf{M}\mathbf{u}_i . \tag{6}$$

From the definition of the generalized eigenvalue, we know that $\mathbf{u}_i^T \mathbf{M} \mathbf{u}_i = 1$ and $(\mathbf{K} - \lambda_i \mathbf{M})\mathbf{u}_i = 0$. Zero coefficient $(\mathbf{u}_i^T (\mathbf{K} - \lambda_i \mathbf{M}) = 0)$ automatically nullifies the gradient of the eigenvector. As a result, we can rearrange the equation and obtain:

$$\partial \lambda_i = \mathbf{u}_i^T \left(\partial \mathbf{K} - \lambda_i \partial \mathbf{M} \right) \mathbf{u}_i \ . \tag{7}$$

Now, we establish a connection between the gradient of vibration frequencies and the gradient of the mass and stiffness matrices.

3.4 Loss Function for Optimization

At this stage, we can optimize the material properties and geometry of the object using target eigenvalues. This optimization is performed by employing the loss function defined as:

$$L_i = ||\lambda_i^{pred} - \lambda_i^{gt}||_1 \ , \tag{8}$$

where λ_i^{gt} is the ground truth eigenvalue of mode i and λ_i^{pred} denotes the predicted eigenvalue. Note that the absence of a loss function regarding eigenvectors does not imply that the eigenvectors will not change. In fact, during the optimization process, gradients are transferred to the matrices $\mathbf{K}$ and $\mathbf{M}$. As these matrices undergo updates through gradient descent, the eigenvectors associated with them naturally evolve as well. This indicates a dynamic change in the eigenvectors even in the absence of direct modifications through the loss function.

For generality, we proceed to compute the predicted sound signal from the predicted eigenvalues as detailed in Sec. 3.4.1. Subsequently, we utilize a hybrid loss function to calculate the loss of the sound signal as detailed in Sec. 3.4.2.

3.4.1 Differentiable Additive Synthesizer. The sound produced by a rigid-body object can be effectively modeled as a bank of damping sinusoidal oscillators. For the i-th mode, denoting its damping factor as d_i and its amplitude as A_i, its frequency can be obtained by:

$$f_i = \frac{\sqrt{\lambda_i - d_i^2}}{2\pi} \ . \tag{9}$$

Let h be the time step size, the sound signal $s_i(n)$ over discrete time steps, n, can be computed as:

$$s_i(n) = A_i e^{-d_i n h} \sin(2\pi f_i n h) \ . \tag{10}$$

Finally, the sound is produced by summing the sound signals for all modes. It is important to note that amplitudes and damping factors are designed to be learned from ground truth data, and amplitudes can implicitly include the acoustic transfer function [James 2016]. Additionally, the eigenvalues λ_i play a crucial role in connecting the sound signal to the physical properties of the object. The computations defined in Equations 9 and 10 are evaluated in parallel along both the time and mode dimensions using PyTorch, enabling automatic differentiation.

When dealing with naturally recorded sounds that contain noise, we enhance the output of the additive synthesizer by combining it with noise filtered by an LTV-FIR filter [Engel et al. 2020]. The parameters of this filter are also learnable, enabling it to adapt to real-world noise characteristics.

3.4.2 Hybrid Loss Function. As suggested in previous differential audio synthesizers [Clarke et al. 2021; Engel et al. 2020], a multi-scale spectral loss is effective for measuring the difference between two audio signals. Given the ground truth and the predicted sound signals, we compute their spectrograms S_i and $\hat{S}_i$, respectively, using a specified FFT size i. The loss is then defined as the sum of the L1 difference between S_i and $\hat{S}_i$, as well as the L1 difference between their respective log spectrograms:

$$L_i = ||S_i - \hat{S}_i||_1 + ||\log S_i - \log \hat{S}_i||_1 \ . \tag{11}$$

The total reconstruction loss is the sum of all the spectral losses with different FFT sizes, which provide varying frequency and temporal resolutions.

Traditional L1 or L2 loss can result in difficult convergence when the initial and ground truth object locations or frequencies significantly differ [Xing et al. 2022]. This issue also arises in differentiable sound rendering. For instance, if the initial frequency far deviates from the ground truth frequency, there may be no overlapping pixels in the spectrogram between the initial mode and the target mode, causing the L1 or L2 loss to yield zero gradients and potentially leading to undesired local minima.

To address this issue, we first treat the spectrogram value in each frequency bin as a high-dimensional point. To measure the distance between the ground truth and the predicted spectrograms, we utilize the optimal transport (Wasserstein) distance. This distance metric considers the cost of moving mass from one distribution to another. In our context, we define the unit moving cost from one frequency bin to another as their corresponding point distance. For efficiency, we employ an efficient algorithm for approximating optimal transport distances using Sinkhorn divergences [Feydy et al. 2019].

As the optimal transport-based loss tends to be less effective when the initial and target spectrograms are already well-aligned, we thus propose to use a hybrid strategy: We train the model using the transport-based loss function until a plateau in loss reduction is observed, indicating no significant further decrease. At this point, we transition to employing the spectral loss function. This staged approach ensures that the model optimally benefits from each type of loss during different phases of the training process.

4 INVERSE RENDERING TASKS AND EXPERIMENTS

We define three types of inference tasks and conduct corresponding experiments to showcase the capability of our differentiable rendering framework. First, we perform an ablation study on the loss function to validate our approach (Sec. 4.1). Next, we reason about the material attributes (Sec. 4.2), geometric shape (Sec. 4.3), and impact position (Sec. 4.4) of the object in a contact event. Please refer to the supplementary video for the demo results of our experiments.

The real-world object data used in the experiments is sourced from the ObjectFolder Real dataset [Gao et al. 2023], which contains multisensory data collected from 100 real-world household objects. The data for each object includes its high-quality 3D mesh, impact sound recordings, and the accompanying video footage for each impact. Our DiffSound is implemented in PyTorch and utilizes the Adam optimizer for optimization.

4.1 Ablation Study on Loss Functions

We first conduct an ablation study to validate the effectiveness of the hybrid loss function compared to either using a single multi-scale L1 loss or a single optimal transport-based loss.

We set up a simple case where the predicted eigenvalues can only be changed proportionally through a trainable scaling factor. We aim to optimize this scaling factor from an initial value of 1.0 to a predefined target value. We select four meshes from the dataset and manually set the material parameters, following the guidelines presented in [James 2016].

As depicted in Figure 4, the results indicate that the optimal transport-based loss shows high effectiveness for optimizing from a bad initial state where the multi-scale L1 loss cannot work. Additionally, our hybrid loss function achieves the best performance compared to either single loss function in all experiments.

4.2 Material Attribute Inference

In this task, we aim to infer the material parameters from the impact sound of an object, assuming the object's geometric model is known.

First, we focus on estimating the damping curve, which is a crucial part of our differentiable additive synthesizer, denoted as S_{θ_S} (see Figure 6). We define the ground truth audios as a_{gt} and use a hybrid loss function, L, for optimization. For each frequency in a set of randomly selected modes, denoted as f_{rand}, our synthesizer assigns a unique damping coefficient. It then uses these coefficients to synthesize the sound for these modes. The optimization of the damping curve can be formulated as:

$$\theta_S^* = \text{argmin}_{\theta_S}(L(S_{\theta_S}(f_{rand}), a_{gt})) \ . \tag{12}$$

After optimization, we remove modes with small amplitudes and interpolate the damping coefficients of the remaining modes to form a continuous damping curve.

Next, we estimate the material parameters of our FEM matrix assembler, A_{θ_A}. These parameters include the ratio of Young's modulus to density (denoted as $\hat{E}$) and Poisson's ratio (denoted as ν). Our framework F_θ integrates several components: a fixed tetrahedron mesh generator G, the FEM matrix assembler A_{θ_A}, an eigenvalue decomposer D, and the previously trained and now fixed additive synthesizer $S_{\theta_S^*}$. For a given input model m, the optimization of the material parameters can be expressed as:

$$\theta_A^* = \text{argmin}_{\theta_A}(L(F_\theta(m), a_{gt})) \ . \tag{13}$$

We separate the training processes for the damping factor and material parameters. This is because the errors in the damping factor and material parameters can influence each other, making optimization challenging. In this two-step process, we sequentially refine the estimation of the damping and material parameters, leading to a more accurate and stable optimization. Figure 7 presents a comparison of the the outcomes from training the damping factor and material parameters separately versus training them simultaneously on a test object. The results demonstrate that our approach of separate training substantially outperforms the simultaneous training method, proving to be highly effective.

Contrasting with prior work [Ren et al. 2013] that used first-order FEM with a fixed Poisson's ratio to predict only Young's modulus, their model's eigenvalues are simplified with a linear proportional relationship with Young's modulus. In contrast, our method optimizes both Young's modulus and Poisson's ratio, addressing the inaccuracies arising from the oversimplified assumption. We provide a detailed comparison with various baselines in Table 1.

Table 1: Material estimation with 16 objects. Baseline 1 [Ren et al. 2013] uses fixed Poisson's ratio and first-order FEM; Baseline 2 applies 2nd-order FEM with fixed Poisson's ratio; Baseline 3 employs 1st-order FEM with learnable Poisson's ratio. Our method consistently outperforms all baselines in relative errors.

	FEM order	Learnable ν	$\hat{E}$ Err.	ν Err.	Spec. Err.
baseline 1	1	✗	0.51	0.68	26.43
baseline 2	2	✗	0.10	0.68	11.21
baseline 3	1	✓	0.51	0.66	27.00
DiffSound	2	✓	**0.07**	**0.26**	**7.95**

We leverage ground-truth audio data, synthesized using second-order FEM for 16 objects with randomly chosen material parameters from a feasible range. Beginning with random initial material parameters, our framework differentiably synthesizes sound by optimizing these parameters to minimize the loss against the ground truth. Moreover, the effectiveness of our approach is evaluated using data from two real-world ceramic objects.

We use the relative error as a metric for $\hat{E}$, ν, and sound spectrogram, defined as $l = \frac{||g-p||_2}{||g||_2}$ for the ground-truth g and the prediction p. We present the quantitative results in Table 1 for synthetic data, along with qualitative examples for real-world data in Figure 8. Our DiffSound demonstrates substantial improvements over all baselines across all metrics, showcasing high effectiveness even in real-world data.

Implementation details. To estimate the damping curve, we initially train an additive synthesizer using target audios composed of 256 modes with filtered noise. The training spans over 10,000 steps in few minutes. Modes with damping coefficients less than 100 are deemed invalid, and the coefficients of the remaining valid modes are linearly interpolated to form a damping curve. Subsequently, we train the material parameters based on the damping curve, spanning 10,000 steps at a learning rate of 0.01, with a transition in the loss function at the 5,000-step mark. Using a background tetrahedral mesh grid resolution of 32^3, the total training duration for each object is approximately 3 hours.

4.3 Shape Attribute Inference

Determining the shape from sound is challenging because different shapes can produce similar sounds upon impact [Kac 1966]. Therefore, we regulate the material coefficients and impose certain geometry constraints to ensure a reliable optimization process.

4.3.1 Geometric Shape Inference. In this task, our focus is on detailed geometry recovery under certain constraints. Our goal is to precisely reconstruct the finer details of shapes from a given coarse voxel grid by utilizing modal sound for inference.

To accomplish this task, we infer the geometric shape from the eigenvalues of vibration modes, which are directly related to

frequencies (Equation 9). Additionally, we constrain the tetrahedral mesh during optimization using a coarse voxel grid. Specifically, we query the SDF values from the MLP and ensure that the SDF of grid points inside the mesh is negative, while those outside are positive. This is enforced using a loss defined as the sum of absolute SDF values of those points whose SDF sign differs from the expected sign. The loss for sound constraint is defined as the L1 loss between the ground truth eigenvalues and the predicted eigenvalues of the first k modes, divided by the norm of the ground truth.

In our framework, the key learnable parameter, denoted as θ, is the implicit SDF representation, SDF_θ, in the tetrahedron mesh generator G_{θ_G}. This generator processes the coordinates of a point to output its SDF value for further synthesis. Our framework integrates the tetrahedron mesh generator G_{θ_G}, the fixed FEM matrix assembler A, and the eigenvalue decomposer D, yielding the function F_θ. This function takes an initial coarse mesh m as input and outputs the first k smallest eigenvalues associated with it.

Let λ_{gt} represent the ground truth eigenvalues and SDF denote the SDF field defined by the initial coarse voxels. The weight of the coarse voxel grid constraint is given by w. For all vertices v in the background grid B, the optimization process can be formulated to minimize the combination of the eigenvalue loss and the SDF sign discrepancy loss:

$$\theta^* = \text{argmin}_\theta(L_{\text{eigen}}(\theta) + w \cdot L_{\text{SDF}}(\theta)) \ . \tag{14}$$

The eigenvalue loss $L_{\text{eigen}}(\theta)$ is defined as:

$$L_{\text{eigen}}(\theta) = \|F_\theta(m) - \lambda_{gt}\|_1 \ . \tag{15}$$

The SDF sign discrepancy loss $L_{\text{SDF}}(\theta)$ is defined over the background grid B to penalize the discrepancy in the signs of the SDF values between the predicted and initial coarse voxel fields as:

$$L_{\text{SDF}}(\theta) = \sum_{v \in B} |SDF_\theta(v)| \cdot |\text{sign}(SDF_\theta(v)) - \text{sign}(SDF(v))| \tag{16}$$

In this formulation, θ^* is the optimized parameter set that minimizes the combined loss.

In our experiments, we generate synthetic data for three objects from Crane et al. [2013] with a ceramic material parameter. We conduct separate experiments for each object and constraint mode number. The geometric shape can be successfully recovered from impact sound and a coarse voxel constraint (16^3), as illustrated by the quantitative results in Figure 9. This capability compensates for the loss of such details in the initial coarse mesh. When applied to coarser voxel constraints (8^3), our approach can still synthesize geometric details that are visually close to the ground truth. The high quality of shape estimation of our approach can also be validated in the accompanying video.

Implementation details. First, we obtain the initial MLP representation for the SDF by optimizing with only the constraint of a coarse voxel grid. The SDF constraint involves 32^3 uniformly distributed points within a cube that is 1.1 times the size of the object's bounding box. This initial training process requires only a few seconds for 1000 steps. Subsequently, we refine the MLP representation by training according to Equation 15 with $\omega = 0.001$. Using a background tetrahedral mesh grid resolution of 32^3, this process spans 100 steps and takes approximately 2 minutes.

4.3.2 Volumetric Thickness Inference. An interesting question arises: Is a given object solid or hollow? And if so, how thick is the inside? In this task, our focus shifts to understanding the internal conditions of an object as a whole. Our goal is to precisely estimate the volumetric thickness of a given object by utilizing modal sound for inference.

To complete this task, we generate hollow meshes from each solid input, varying only in thickness, and then predict their thickness based on their eigenvalues. Mesh "thickness" is defined using the solid mesh's Signed Distance Field (SDF): the smallest SDF value, s_{min}, corresponds to the farthest internal point from the outer surface. An object of thickness t includes points within $t \cdot -(s_{\text{min}})$ from this surface. Thus, a point P lies inside an object of thickness t if its SDF value $SDF(P)$ satisfies $-t \cdot s_{\text{min}} < SDF(P) < 0$.

Based on the defined "thickness", we enhanced our tetrahedral mesh generator $G_{\theta_G} = G_t$ with a thickness parameter t, enabling it to synthesize hollow voxel meshes of thickness t from a solid mesh. We chose several target thicknesses t_{target} for each input mesh and created corresponding ground truth meshes for each. During optimization, G_t differentiably generates a hollow voxel mesh from the initial SDF of the mesh, matching the current thickness t. We then compute the L2 loss between the first k eigenvalues of the current and ground truth meshes, updating t based on its gradient.

We consolidate the tetrahedron mesh generator G_t, the fixed FEM matrix assembler A and the eigenvalue decomposer D in our framework into a function F_t, which takes an input mesh and outputs first k eigenvalues of its voxel corresponding tetrahedron mesh with thickness t. Assuming ground truth eigenvalues is λ_{gt}, for initial solid mesh m, formulation of this optimization process can be expressed as:

$$t^* = \text{argmin}_t(||F_t(m) - \lambda_{gt}||^2) \ . \tag{17}$$

In our experiment, we selected four models with varying materials. We generated target grids for each model at thicknesses of 0.3, 0.4, 0.5, 0.6, and 0.7, and used their first $k = 32$ eigenvalues as ground truth. We observed that thicknesses below 0.3 led to holes in the mesh due to limited resolution, while thicknesses above 0.7 yielded eigenvalues akin to those of solid objects. The results, detailed in Table 2, indicate our method's proficiency with smooth, thick meshes, but potential inaccuracies with complex surfaces or thinner meshes. This can be attributed to a more complex nonlinear relationship between eigenvalues and thickness in such cases.

Implementation details. Using a background tetrahedral mesh grid resolution of 64^3, the training process encompasses 500 steps at a learning rate of 0.02, taking approximately 2 hours per object and target thickness.

4.3.3 Shape Morphing Inference. In this task, we focus on two distinct objects and a series of intermediate morphing shapes between them. Our objective is to accurately identify the specific shapes within this progression, utilizing modal sound for inference. To accomplish this task, we begin by calculating the SDF values for each vertex in the DMTet background grid of the two initial meshes. We then interpolate the SDF values from these meshes for each vertex. Let v_{SDF1} and v_{SDF2} be the SDF values for a background grid vertex v relative to the two models, and t be the interpolation coefficient. The vertex's interpolated SDF value is

Table 2: Volumetric thickness inference using synthetic data tested with different objects. We measure the prediction error against the ground-truth coefficient using MAE. Our approach achieves high prediction accuracy.

object	target thickness					MAE
	0.3	0.4	0.5	0.6	0.7	
Bunny	0.304	0.407	0.508	0.608	0.709	0.0073
Armadillo	0.338	0.456	0.590	0.696	0.730	0.0623
Bulbasaur	0.308	0.411	0.512	0.614	0.718	0.0125
Squirtle	0.312	0.416	0.520	0.624	0.718	0.0177

$v_{SDF} = t \cdot v_{SDF1} + (1-t) \cdot v_{SDF2}$. As t varies from 0 to 1, the mesh transitions from the first to the second mesh, as depicted in the Target part in Figure 10. Our objective is to find the mesh that optimally fits the target sound during this morphing, by optimizing the interpolation coefficient t.

For two initial meshes m_1 and m_2, $SDF(\cdot)$ gives its SDF values in all background grid vertexes $SDF(m_1)$ and $SDF(m_2)$. We consolidate the tetrahedron mesh generator $G_{\theta_G} = G_t$, the fixed FEM matrix assembler A, and the eigenvalue decomposer D in our framework into a function F_t, which takes background grid SDF and outputs its corresponding first k eigenvalues.

Assuming ground truth first k eigenvalues is λ_{gt}, The formulation of this optimization process can be expressed as:

$$t^* = \text{argmin}_t(||F_t(t \cdot SDF(m_1) + (1-t) \cdot SDF(m_2)) - \lambda_{gt}||^2). \quad (18)$$

To validate the feasibility of our framework for this task, we select a series of initial mesh pairs, select a range of target interpolation coefficients, interpolate SDF values from target coefficients, synthesize morphed tetrahedral voxel meshes from SDF values, and extract its first k eigenvalues as ground truth. During the optimization process, we start from randomly initialized interpolation coefficients t, synthesize the interpolated meshes from current interpolation coefficients, calculate the L2 loss between their eigenvalues and the ground truth, and backpropagate gradients to update the interpolation coefficients t.

In our experiment, we used Squirtle and Bulbasaur, and Bunny and Spot, as two pairs of initial meshes for morphing. We interpolated models using SDF coefficients of 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0, and extracted their first $k = 32$ eigenvalues as shape morphing targets. To keep the interpolation coefficient t between 0 and 1, t was represented as a weighted sum of 16 evenly spaced numbers from 0 to 1. These weights, treated as learnable parameters, were normalized to be positive and sum to 1. Figure 10 visualizes the results, showing our framework's efficiency in finding shapes that align closely with the target eigenvalues during transformation. The implementation details are exactly the same as those for the experiments on thickness inference except for the learning rate, which is set to be 0.05.

4.4 Impact Event Inference

Impact position and amplitude are not explicitly optimized as a learnable parameter. However, the learnable mode amplitude A in Equation 10 implicitly encodes information about the impact position and amplitude. In other words, the mode amplitude can be utilized to predict the object's impact position and amplitude, a conclusion that has also been reached in Van den Doel and Pai [1998].

In this task, we aim to infer the impact position from the recorded sound, given that the object's mesh is known. First, we optimize the material parameters from sound following the process outlined in Sec. 4.2. Simultaneously, we optimize the amplitudes of all modes, denoted as $\mathbf{A} = [A_0, A_1, ..., A_n]$. Then, using the estimated material parameters, we apply forward modal sound simulation, which includes acoustic transfer [James et al. 2006], to obtain the simulated amplitudes of all modes $\hat{\mathbf{A}}_i$ when impacting each mesh vertex v_i. We measure the likelihood that the impact position corresponding to the recorded sound is near vertex v_i by evaluating the similarity between $\mathbf{A}$ and $\hat{\mathbf{A}}_i$. We choose recorded real data of a ceramic bowl from the ObjectFolder Real dataset [Gao et al. 2022] and use cosine similarity to compute the surface likelihood distribution, as visualized in Figure 5. In this example, the bowl exhibits rotational symmetry around its central axis. Theoretically, striking any points on the bowl that are rotationally symmetric about this axis (forming a circle centered on the axis) would produce identical sounds with symmetric impulse responses. The impact positions predicted by the mode amplitudes are deemed reasonable and accurate if they fall within the same circle as the groundtruth (i.e., they are rotationally symmetric about the central axis). For ease of visual comparison, we rotate the probability heatmap of the predicted impact positions to align them with the groundtruth. And our method predicts a high likelihood around the ground truth impact position.

5 CONCLUSION

We presented a differentiable modal sound rendering framework that enables inverse rendering by computing the gradient of the simulation function with respect to input physical parameters (e.g., material parameters). We have verified the effectiveness of our loss strategy with ablation experiments and demonstrated the generality and diversity of DiffSound for the inverse rendering of material parameters, impact positions, and the shape of the sounding objects. We hope our framework can unlock new multisensory applications in the fields of robotics and embodied AI.

Nonetheless, our framework currently faces several challenges. These include difficulties in handling complex shapes, particularly thin shells, and limitations in accurately modeling heavily nonlinear sounds. Additionally, if the distribution of an object's damping coefficients fluctuates significantly from low to high frequencies, a simple interpolated damping curve may fail to accurately represent the true distribution. Furthermore, optimizing the rendering speed to support real-time applications remains a critical priority. In future endeavors, we aim to develop a more comprehensive and efficient differentiable sound rendering framework, building upon the foundation laid by this work.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their helpful suggestions. This work is supported by the National Key R&D Program of China (No. 2022YFB3303403) and NSFC of China (No. 62172013).

REFERENCES

Adam W. Bargteil and Elaine Cohen. 2014. Animation of Deformable Bodies with Quadratic BéZier Finite Elements. *ACM Trans. Graph.* 33, 3, Article 27 (jun 2014), 10 pages.

Gaurav Bharaj, David I. W. Levin, James Tompkin, Yun Fei, Hanspeter Pfister, Wojciech Matusik, and Changxi Zheng. 2015. Computational Design of Metallophone Contact Sounds. *ACM Trans. Graph.* 34, 6, Article 223 (nov 2015), 13 pages.

Jeffrey N. Chadwick, Changxi Zheng, and Doug L. James. 2012. Precomputed Acceleration Noise for Improved Rigid-Body Sound. *ACM Trans. Graph.* 31, 4, Article 103 (jul 2012), 9 pages.

Samuel Clarke, Ruohan Gao, Mason Wang, Mark Rau, Julia Xu, Mark Rau, Jui-Hsien Wang, Doug James, and Jiajun Wu. 2023. RealImpact: A Dataset of Impact Sound Fields for Real Objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Samuel Clarke, Negin Heravi, Mark Rau, Ruohan Gao, Jiajun Wu, Doug James, and Jeannette Bohg. 2021. DiffImpact: Differentiable Rendering and Identification of Impact Sounds. In *5th Annual Conference on Robot Learning*.

Simon Le Cleac'h, Hong-Xing Yu, Michelle Guo, Taylor A. Howell, Ruohan Gao, Jiajun Wu, Zachary Manchester, and Mac Schwager. 2023. Differentiable Physics Simulation of Dynamics-Augmented Neural Objects. *Robotics and Automation Letters (RA-L)* (2023).

COMSOL AB, Stockholm, Sweden. 2005. *Comsol multiphysics user's guide.*

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. 2018. End-to-End Differentiable Physics for Learning and Control. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc.

Jonas Degrave, Michiel Hermans, Joni Dambre, et al. 2019. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics* (2019), 6.

Akio Doi and Akio Koide. 1991. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems* 74, 1 (1991), 214–224.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable Projective Dynamics. *ACM Trans. Graph.* 41, 2, Article 13 (nov 2021), 21 pages.

Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. 2020. DDSP: Differentiable Digital Signal Processing. In *International Conference on Learning Representations*.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. 2019. Interpolating between Optimal Transport and MMD using Sinkhorn Divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*. 2681–2690.

Ruohan Gao, Yen-Yu Chang, Shivani Mall, Li Fei-Fei, and Jiajun Wu. 2021. ObjectFolder: A Dataset of Objects with Implicit Visual, Auditory, and Tactile Representations. In *5th Annual Conference on Robot Learning*.

Ruohan Gao, Yiming Dou, Hao Li, Tanmay Agarwal, Jeannette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. 2023. The ObjectFolder Benchmark: Multisensory Object-Centric Learning with Neural and Real Objects. In *CVPR*.

Ruohan Gao, Zilin Si, Yen-Yu Chang, Samuel Clarke, Jeannette Bohg, Li Fei-Fei, Wenzhen Yuan, and Jiajun Wu. 2022. ObjectFolder 2.0: A Multisensory Object Dataset for Sim2Real Transfer. In *CVPR*.

Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: Analytically Differentiable Dynamics for Multi-Body Systems with Frictional Contact. *ACM Trans. Graph.* 39, 6, Article 190 (nov 2020), 15 pages.

David Hahn, Pol Banzet, James M. Bern, and Stelian Coros. 2019. Real2Sim: Visco-Elastic Parameter Estimation from Dynamic Motion. *ACM Trans. Graph.* 38, 6, Article 236 (nov 2019), 13 pages.

Philipp Holl, Nils Thuerey, and Vladlen Koltun. 2020. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations*.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. *ICLR* (2020).

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics. In *2019 International Conference on Robotics and Automation (ICRA)* (Montreal, QC, Canada). IEEE Press, 6265–6271.

Thomas JR Hughes. 2012. *The finite element method: linear static and dynamic finite element analysis.* Courier Corporation.

Doug L. James. 2016. Physically Based Sound for Computer Animation and Virtual Environments. In *ACM SIGGRAPH 2016 Courses* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 22, 8 pages.

Doug L James, Jernej Barbič, and Dinesh K Pai. 2006. Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 987–995.

Xutong Jin, Sheng Li, Tianshu Qu, Dinesh Manocha, and Guoping Wang. 2020. Deep-Modal: Real-Time Impact Sound Synthesis for Arbitrary Shapes. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 1171–1179.

Xutong Jin, Sheng Li, Guoping Wang, and Dinesh Manocha. 2022. NeuralSound: Learning-Based Modal Sound Synthesis with Acoustic Transfer. *ACM Trans. Graph.* 41, 4, Article 121 (jul 2022), 15 pages.

Mark Kac. 1966. Can one hear the shape of a drum? *The american mathematical monthly* 73, 4P2 (1966), 1–23.

Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A. Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. 2022b. See, Hear, and Feel: Smart Sensory Fusion for Robotic Manipulation. In *CoRL*.

Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022a. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. *ACM Trans. Graph.* 42, 1, Article 2 (oct 2022), 20 pages.

Junbang Liang, Ming Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc.

Andreas Longva, Fabian Löschner, Tassilo Kugelstadt, José Antonio Fernández-Fernández, and Jan Bender. 2020. Higher-Order Finite Elements for Embedded Simulation. *ACM Trans. Graph.* 39, 6, Article 181 (nov 2020), 14 pages.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid Control Using the Adjoint Method. 23, 3 (aug 2004), 449–456.

Johannes Mezger, Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2008. Interactive Physically-Based Shape Editing. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (Stony Brook, New York) *(SPM '08)*. Association for Computing Machinery, New York, NY, USA, 79–89.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8280–8290.

J. Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. 2021. gradSim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*.

James F. O'Brien, Chen Shen, and Christine M. Gatchalian. 2002. Synthesizing Sounds from Rigid-Body Simulations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. Association for Computing Machinery, New York, NY, USA, 175–181.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

Jovan Popović, Steven M. Seitz, and Michael Erdmann. 2003. Motion Sketching for Control of Rigid-Body Simulations. *ACM Trans. Graph.* 22, 4 (oct 2003), 1034–1054.

Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. 2020. Scalable Differentiable Physics for Learning and Control. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 727, 10 pages.

Nikunj Raghuvanshi and Ming C. Lin. 2006. Interactive Sound Synthesis for Large Scale Environments. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (Redwood City, California) *(I3D '06)*. Association for Computing Machinery, New York, NY, USA, 101–108.

Zhimin Ren, Hengchin Yeh, and Ming C. Lin. 2013. Example-Guided Physically Based Modal Sound Synthesis. *ACM Trans. Graph.* 32, 1, Article 1 (feb 2013), 16 pages.

Connor Schenck and Dieter Fox. 2018. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*. PMLR, 317–335.

Teseo Schneider, Jérémie Dumas, Xifeng Gao, Mario Botsch, Daniele Panozzo, and Denis Zorin. 2019. Poly-spline finite-element method. *ACM Transactions on Graphics (TOG)* 38, 3 (2019), 1–16.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 6087–6101.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses* (Los Angeles, California) *(SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 20, 50 pages.

A. Sterling, N. Rewkowski, R. L. Klatzky, and M. C. Lin. 2019. Audio-Material Reconstruction for Virtualized Reality Using a Probabilistic Damping Model. *IEEE Transactions on Visualization and Computer Graphics* (2019), 1–1. https://doi.org/10.1109/TVCG.2019.2898822

Marc Toussaint, Kelsey R. Allen, Kevin A. Smith, and Joshua B. Tenenbaum. 2019. Differentiable Physics and Stable Modes for Tool-Use and Manipulation Planning - Extended Abtract. In *Proceedings of the Twenty-Eighth International Joint Conference*

on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization, 6231–6235.

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe Control of Smoke Simulations. *ACM Trans. Graph.* 22, 3 (jul 2003), 716–723.

Kees van den Doel, Paul G. Kry, and Dinesh K. Pai. 2001. FoleyAutomatic: Physically-Based Sound Effects for Interactive Simulation and Animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 537–544.

Kees Van den Doel and Dinesh K Pai. 1998. The sounds of physical shapes. *Presence* 7, 4 (1998), 382–395.

Chris Wojtan, Peter J. Mucha, and Greg Turk. 2006. Keyframe Control of Complex Particle Systems Using the Adjoint Method. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vienna, Austria) *(SCA '06)*. Eurographics Association, Goslar, DEU, 15–23.

Jiankai Xing, Fujun Luan, Ling-Qi Yan, Xuejun Hu, Houde Qian, and Kun Xu. 2022. Differentiable Rendering Using RGBXY Derivatives and Optimal Transport. *ACM Trans. Graph.* 41, 6, Article 189 (nov 2022), 13 pages.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. 2021. An End-to-End Differentiable Framework for Contact-Aware Robot Design. In *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh (Eds.). https://doi.org/10.15607/RSS.2021.XVII.008

Zhoutong Zhang, Qiujia Li, Zhengjia Huang, Jiajun Wu, Joshua B. Tenenbaum, and William T. Freeman. 2017. Shape and Material from Sound. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1278–1288.

Changxi Zheng and Doug L. James. 2011. Toward High-Quality Modal Contact Sound. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 38, 12 pages.

Bofang Zhu. 2018. The finite element method: fundamentals and applications in civil, hydraulic, mechanical and aeronautical engineering. (2018).

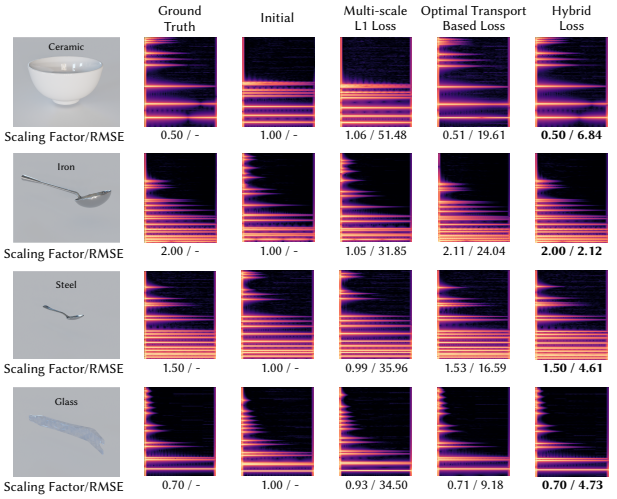

Figure 4: Ablation study on loss functions. We show the spectrograms, scaling factors of eigenvalues, and RMSE in different setups. Across all setups, our hybrid loss function consistently outperforms the one using only the multi-scale L1 loss or optimal transport-based loss.

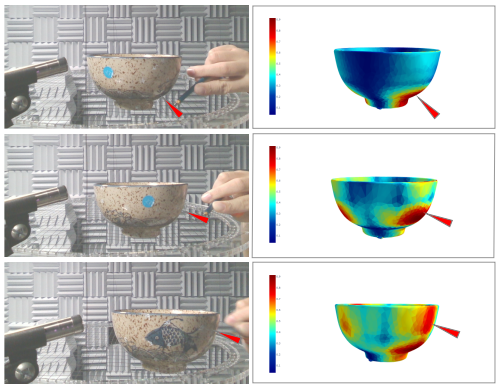

Figure 5: Visualization of the surface likelihood distribution (probability heatmap) of the impact position on the object's surface for an example object. The predicted positions are considered reasonable and accurate if they fall within the region that is rotationally symmetric about the central axis relative to the groundtruth.

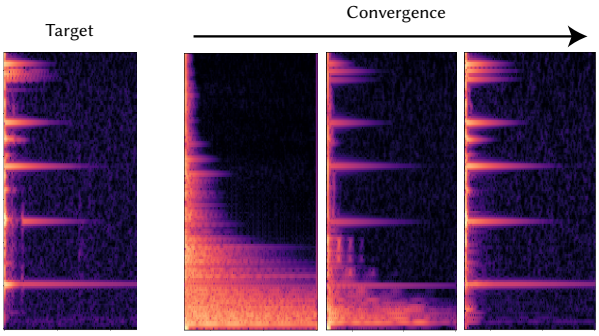

Figure 6: Training process of estimating the damping curve. We utilize 256 initial modes to comprehensively cover all target modes. After training, degraded modes are subsequently removed.

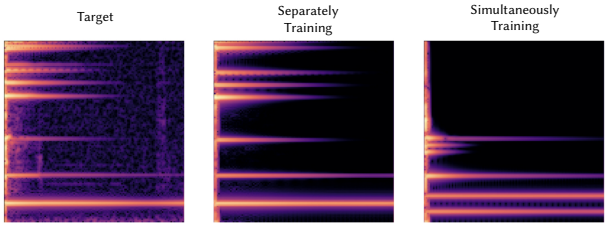

Figure 7: The comparison of outcomes from separately training the damping factor and material parameters versus their simultaneous training for a test object. The images clearly demonstrate that the separate training strategy yields significantly better results than the simultaneous approach, demonstrating the effectiveness of our separate training strategy.

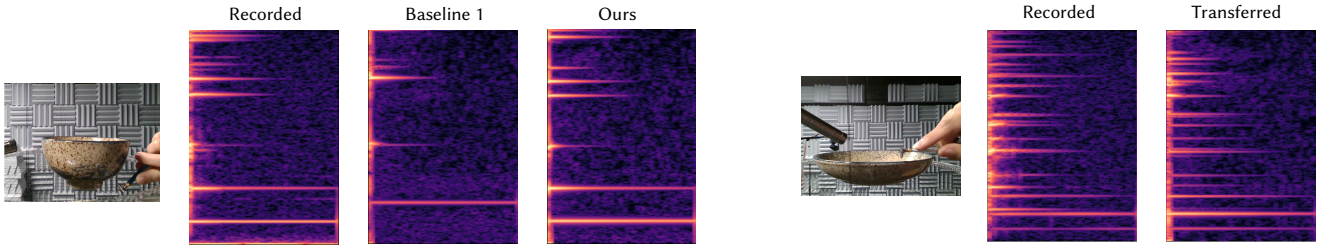

Figure 8: (Left) Material estimation from real-world recorded sound with our DiffSound method and the baseline of [Ren et al. 2013], which uses a fixed value of Poisson's ratio in first-order FEM. (Right) Transfer of the material parameters optimized from a ceramic bowl to a plate with the same material, with additional fine-tuning of the noise filter and mode amplitude.

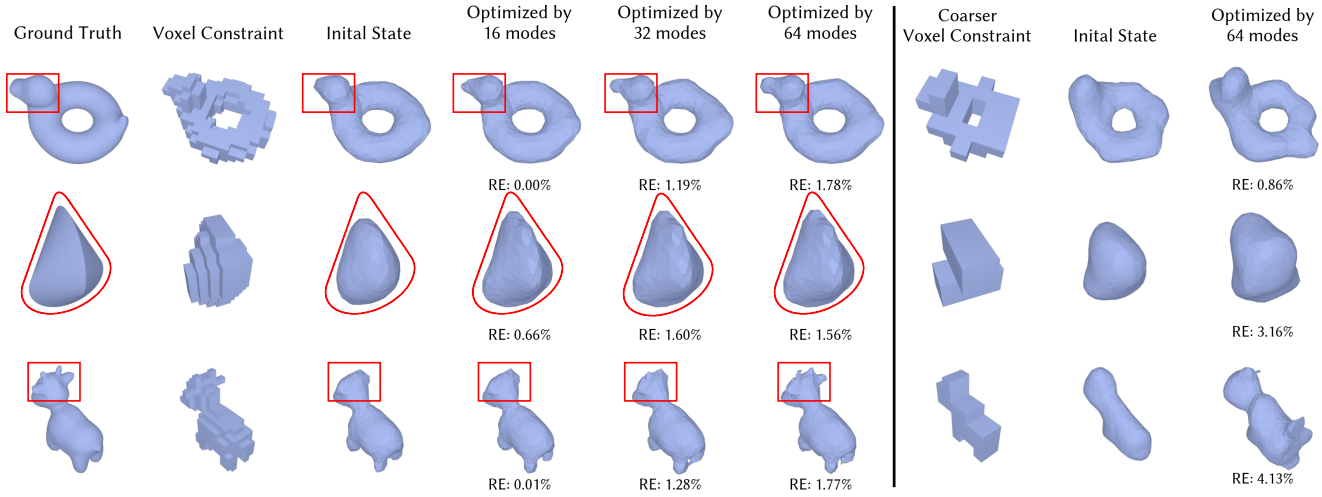

Figure 9: Optimizing shape detail through sound mode (eigenvalues of the ground-truth model) and voxel grid (16^3). Our approach demonstrates its capability to restore shape details with small Relative Error (RE) of eigenvalues. With an increase in the number of modes, fitting all modes simultaneously becomes challenging, causing an increase in RE. However, more modes enforce stricter constraints on shape optimization, yielding an optimized mesh that more closely resembles the ground truth in detail. Three rightmost columns highlight our outcomes when applying coarser voxel constraints (8^3). Our approach can visually synthesize geometric details that closely resemble the ground truth.

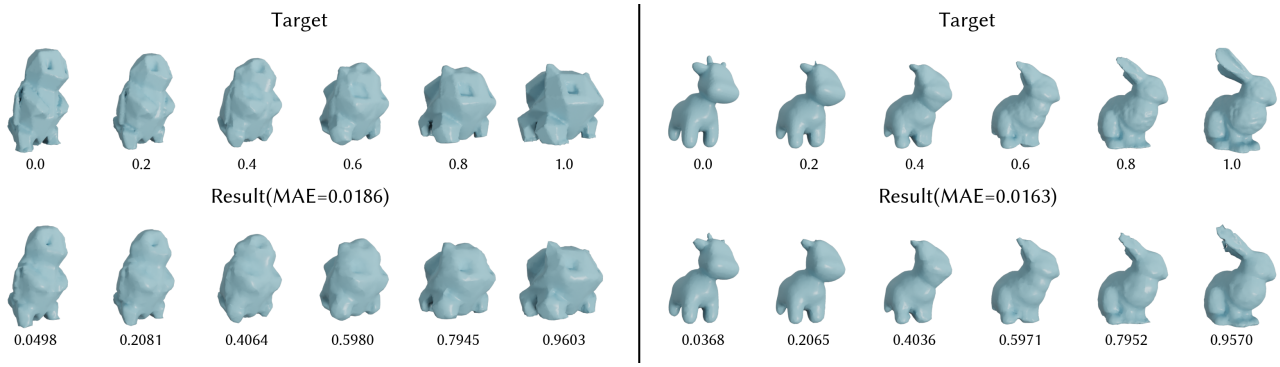

Figure 10: Shape morphing recovery from successive meshes. We annotate each shape with its corresponding target/predicted interpolation coefficients. The prediction accuracy of these coefficients is assessed using Mean Absolute Error (MAE). Our results demonstrate that our framework effectively predicts the mesh nearest to the target eigenvalue, along with its associated shape.