Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)
Ladislav Kavan and Chris Wojtan (Editors)

# Interactive Physically-Based Sound Design of 3D Model using Material Optimization

Kazuhiko Yamamoto[1] Takeo Igarashi[1]

[1]The University of Tokyo

**Abstract**

*Physically-based sound rendering enriches 3D animation. However, it is difficult to make an object with a given shape produce a specific sound using physically-based sound rendering because the user would need to define appropriate internal material distribution. To address this, we propose an example-based method to design physically-based sound for a 3D model. Our system optimizes the material distribution inside the 3D model so that physically-based sound rendering produces sounds similar to the target sounds specified by the user. A problem is that modal analysis required for this optimization is prohibitively expensive. In order to run the optimization at an interactive rate, we present fast approximate modal analysis that enables three orders of magnitude acceleration of the eigenproblem computation compared to standard modal analysis for an elastic object. It consists of data-driven online coarsening of the mesh and hierarchical component mode synthesis with efficient error correction. We demonstrate the feasibility of the method with a set of comparisons and examples.*

Categories and Subject Descriptors (according to ACM CCS): Information Systems [H.5.5]: Sound and Music Computing—Signal Synthesis Computer Graphics [I.3.6]: Methods and Techniques—Interaction Techniques; Mathematics of Computing [G.1.6]: Optimization—;

## 1. Introduction

Realistic sound effects that respond to visual events in a scene significantly enhance the user experience in virtual environments. Traditionally, sound effect designers prepared pre-recorded and pre-edited audio samples, and these samples were synchronized to visual events by manual tweaking (e.g., for feature films) or using scripts (e.g., for VR and games). However, it is laborious to prepare appropriate audio samples for a large variety of visual events, limiting expressiveness and variation of sound effects.

As a solution for this problem, physically-based sound synthesis techniques known as sound rendering [OCE01] have been proposed by the graphics community in the last decade. Modal sound synthesis [Adr91, ZJ11] is widely used for sound simulation of quasi-rigid bodies; it can efficiently produce physically-plausible sounds responding to a large variation of visual events (e.g., collision, bounce, and scratch). This method reduces the effort of manipulating a large number of audio samples for sound designers because it does not use any pre-recorded audio sample. All sounds are automatically triggered and rendered by physical simulation responding to visual events. However, although input parameter for these techniques is the material distribution inside the model, designing the internal material distribution properly so that the system produces a specific sound as a result of physical simulation is difficult even for professional designers.

To address this problem, we propose an example-based interactive design framework for rendering the physically-based sound of a 3D model using material optimization (Figure 1). Our approach enables a user to control the timbre of modal sound synthesis easily without directly specifying the internal material distributions. The user first provides a 3D surface model to the system as input. Next, the user selects a few sample positions on the model surface, and assigns corresponding sound clips that define the target sounds to be rendered when the positions are struck. The system then optimizes the material distribution inside the model so that physically-based sound simulations yield the expected sounds.

However, modal analysis that is required for obtaining the vibrational property of the object at an iteration in our optimization

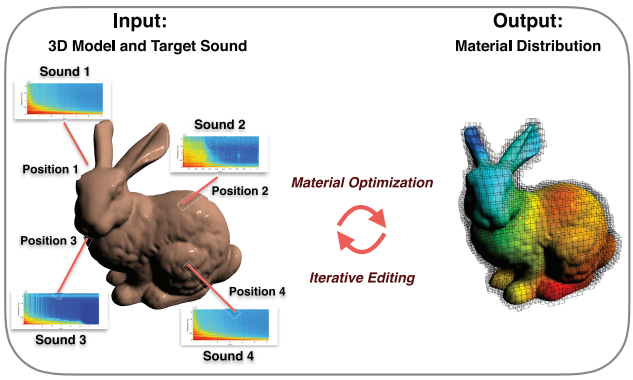


**Figure 1:** *The user assigns several target sounds to sample points on given 3D model as examples. The system then optimizes the material distribution inside the model so that physically-based sound simulation produces the expected sounds. The user can check the simulated sound during the optimization interactively and reassign additional target sounds to design the desired sounding object. Finally, the system outputs embedded FEM mesh with eigenpairs which can be used for standard physically-based sound rendering pipeline.*

is a prohibitively expensive. To execute the optimization at an interactive rate, we present a novel fast approximate modal analysis method that achieves three orders of magnitude acceleration compared to the standard modal analysis (Figure 2). Our technique consists of data-driven finite element coarsening of the mesh and hierarchical component mode synthesis with efficient error correction. Our data-driven online coarsening extends Chen et al.'s method [CLSM15] to handle a large range of continuous material settings by reducing the material parameter space, and can be evaluated with a constant cost for a large amount of datasets using regression forests. Additionally, our highly parallelized hierarchical component mode synthesis extends conventional methods [BC68] to efficiently compute approximate solutions of modal analysis, and our error correction algorithm efficiently improves its accuracy.



DOI: 10.2312/sca.20161240

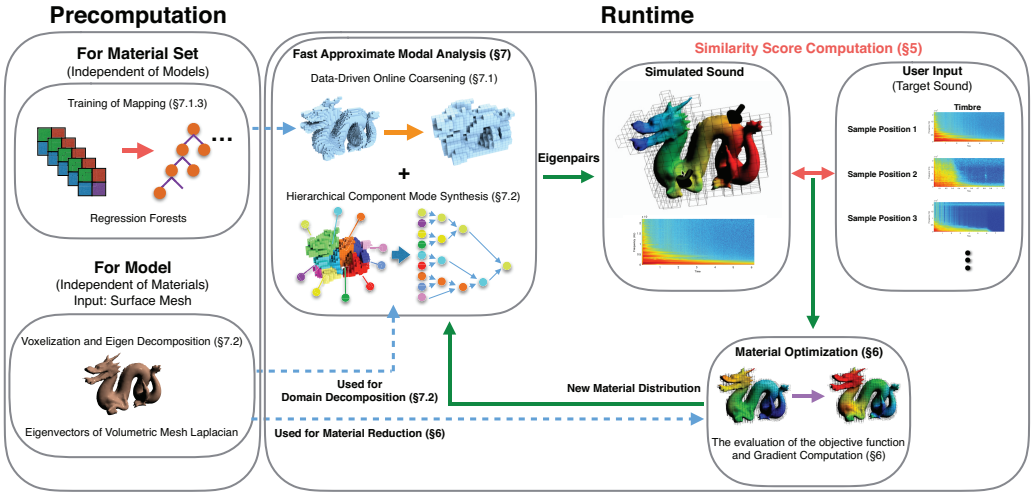


**Figure 2:** *Algorithm Overview. Our optimization algorithm consists of the precomputation and runtime. An iteration of our optimization procedure at runtime consists of 3 steps. First, the system computes modal analysis to obtain the vibrational property of the object. Second, it computes the similarity score between the simulated sounds of the object and user specified target sounds. According to this similarity score, the system updates the material distribution inside the object to minimise the cost.*

## 2. Related Work

**Parameter Acquisition for Modal Sound Synthesis:** To determine the material parameters used in modal sound synthesis, Pai et al. [PvdDJ*01] and Corbett et al. [CvdDLH07] acquired the parameters from actually measured impact sound data, and interpolated them in auditory space. A robotic actuated device is used to apply impulses on a real object at a large number of sample points, and map the recorded impact sounds to virtual objects. However, the measurement procedure of such a huge number of samples for an object and manipulating them are prohibitively expensive. FoleyAutomatic [vdDKP01] also employed similar approach, but interpolated them in modal space for achieving rich sound interactions. However, they also require sufficient amount of samples to estimate the modal function on the surface. Same example sound can be reused at different locations, but it causes a lack of the sound variations when the object interacts with other objects at various locations. This problem becomes profound when the model to be designed has a larger scale.

To avoid measuring such a huge number of parameters for one object from many audio clips, Lloyd et al. [LRG11] proposed a data-driven approach to assign the sound of an object from only one audio clip. They estimated the modal parameters from the audio clip, and at runtime, they randomized the mixture gains of all the tracked modes to generate imaginary varied sounds when hitting different locations on the object. However, this method produces unnatural artifacts because the sounds are not consistent with hit points.

As another approach, Ren et al. [RYL13] proposed a method to estimate the material specific parameter (Rayleigh damping parameters) directly instead of modal parameters from a audio clip under the assumption of uniform material distribution inside the object. The advantage of their approach is that it enables the estimated material parameters to be transferred to different shapes. However, their approach requires that the real object have exactly the same shape as the virtual model to be estimated and should be easy to prepare. These requirements are impractical to implement in actual scenes, which is considered in this paper.

**Vibrational Property Optimization:** To obtain the desired vibrational property of an object, Yamasaki et al. [YNY*10] optimized the shape and topology of an industrial structure using levelset optimization, and controlled the several lowest eigenfrequencies. Yua et al. [YJKK10, YJK13] optimized the topology of a violin's body as specific thin shell structure to control the mode frequencies and amplitudes (mode vectors) that are expected to be largely contributed to the timbre. Bharaj et al. [BLT*15] optimized the shape of a common elastic structure to control both a few mode frequencies as well as their amplitudes for fabricating metal percussion instruments. Our formulation is similar to theirs, but there are four differences. 1: We control a much larger number of modes for dramatically changing the sound's timbre and sacrificing the fabrication possibility. 2: We optimize the material distribution while maintaining the shape whereas they optimize the shape. 3: Our optimization runs at an interactive rate that is enabled by an expansion of data-driven finite elements method (FEM) [CLSM15] and highly parallelized hierarchical component mode synthesis. 4: Our objective function considers the perceptual differences of two sounds whereas they use square distances of frequencies and amplitudes.

**Modal Analysis:** is a well-studied technique in both computer graphics and engineering. It solves the generalized eigenproblem of the finite element stiffness and mass matrices to obtain the vibrational frequencies and the corresponding deformations [HSO03]. Because modal analysis is a time-consuming operation, it is usually used for only the precomputation phase. As some exceptions, Umetani et al. [UMIT10] introduced 2D modal analysis into an interactive design tool for percussion instrument by limiting the fundamental mode computation. Maxwell and Bindel [MB07] computed quasi-3D modal analysis of thin shell structure percussion instruments including the several overtones at a quasi-interactive rate. We introduced 3D modal analysis of a more complex structure into an interactive application.

Many studies focused on the improvement of the computational efficiency of modal analysis. A powerful solution is the domain decomposition approach called the component mode synthesis method (CMS) [Hur65]. CMS decomposes a large problem into many small problems of subdomains and merges them. There are several variations of CMS according to how the boundaries between subdomains are treated [CP88, YVC13]. The major approach is the Craig-Bamptom method [BC68] that treats the interfaces of subdomains as fixed. However, finding an optimal division of a mesh in subdomains is non-trivial, and it should be often

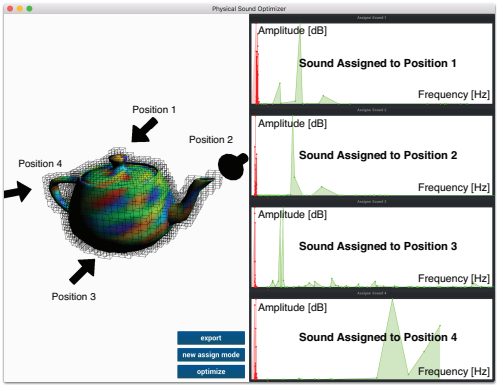


**Figure 3:** *The user interface view. The left pane allows the user to assign the target sounds for the model and preview the contact sound while the right views represent the power spectrums of assigned sounds (green) and the sounds when the positions the user selected are struck (red). The black arrows on the left pane represent the positions the user assigned targe sounds.*

undertaken manually for improving the accuracy. It requires additional expertise and manual efforts by the user. Our approach does not distinguish between subdomains and boundaries, and automatically decomposes it as a hierarchical structure and merges them in parallel. In addition, we improve the accuracy using the fast error correction algorithm, which consists of a combination of the subspace iteration method [Bat13] and sparse mass-Gram-Schmidt process [YLX*15].

## 3. User Workflow

This section describes the user workflow of our interactive physically-based sound design tool. Please see the supplemental video for an interactive demonstration. As seen in the screen capture shown in Figure 3, the user first provides a 3D surface model as an input. The system automatically voxelizes it and converts it into a uniform hexahedral finite element mesh, and executes precomputations as described in the next section. Next, the user selects a vertex position on the surface of the mesh using the mouse, and assigns a sound clip to the position by a drag-and-drop operation. The sound clip defines the sound to be rendered when the model is stuck at the position. The assigned sound clip can be either a pre-recorded real sound (exists in the real world) or an artificial sound (e.g., sound generated by sound synthesizer), but it needs to be an attenuated contact-like sound (free vibrational sound caused by single impulse. An impulse response is ideal). The system allows the user to select multiple positions for each corresponding sound clip. After assigning sounds, the user presses the "optimize" button, and the system optimizes the material distribution inside the model to obtain the desired sound properties. Finally, the system exports the optimized embedded finite element mesh for the surface model with the eigenpairs, and the user can use it for modal sound synthesis.

The optimization gradually progresses at an interactive rate. The system visualizes the current material distribution inside the model by colors and the resulting sounds when the sample positions are struck by power spectrums. The user also can check the sound by clicking the mouse on the mesh surface during optimization at any time. The user can stop the optimization procedure at an arbitrary timing, reassign another sound to a new sample point, and restart the optimization iteratively. In this way, the user can interactively design the physically-based sound for a 3D object as if it were a sound synthesizer.

## 4. Algorithm Overview

Figure 2 shows an overview of our optimization algorithm. Our algorithm consists of two stages: the precomputation stage and the runtime. The precomputation stage consists of two parts. One is precomputation for each material set (independent of models), and it constructs regression forests for data-driven FEM. The regression forests are used for online mesh coarsening using data-driven FEM (§7.1). The other is precomputation for each input model (independent of materials), and it involves voxelizing the model into a hexahedral FEM mesh and computation of the eigenvectors of the volumetric Laplacian of the mesh following [XLCB15]. The eigenvectors of the volumetric Laplacian are used for material reduction (§6), and mesh segmentation (§7.2).

At runtime, the system minimizes the perceptual difference between the user-specified input sound and simulated sound by iterative optimization of material distribution (§6). We consider vibrational property (mode frequencies and amplitudes) to measure perceptual difference (§5). We optimized Young's modulus at each element of FEM, and we kept the densities and Poisson's ratios constant for simplicity. At each iteration, it is necessary to execute a modal analysis of the model to compute the resulting sound. Conventional modal analysis solves the generalized eigenproblem of large stiffness and mass matrices, but it is prohibitively expensive and impractical to use during iterative optimization. To address this, we propose a fast approximate modal analysis based on a combination of data-driven FEM using regression forests (§7.1) and hierarchical component mode synthesis method including error correction (§7.2).

## 5. Problem Formulation

When the user assigns a sound clip onto a sample position, the system extracts the parameters of the sound's timbre from it. An attenuated contact sound can be parameterized by modal parameters (frequencies, amplitudes, and dampings). For the details of the modal parameters, please see Appendix 1. We employ Ren et al.'s technique [RYL13] to extract these parameters from a sound clip. We also extract the residual parameters following them. After $T$ assignments, the system has $N$ sorted mode frequencies of assigned sounds $(F_1,...,F_N)$, corresponding dampings $(D_1,...,D_N)$, corresponding residuals $(R_1,...,R_N)$, and corresponding amplitudes at $T$ sample positions $(A_1^1,...,A_N^1),...,(A_1^T,...,A_N^T)$. We call these extracted parameters as target parameters.

For a given finite element mesh, we compute the first $N$ mode frequencies $(f_1,...,f_N)$ and corresponding amplitudes at $T$ sample positions $(a_1^1,...,a_N^1),...,(a_1^T,...,a_N^T)$ using modal analysis. The modal analysis computes a generalized eigenproblem: $KU = \Lambda MU$, where $K$ and $M$ denote the stiffness and mass matrix respectively and $\Lambda$ and $U$ denote the eigenvalues and the corresponding eigenvectors. To compute the mode amplitudes, we assume each sample position $p_i$ $(i = 1,...,N_p)$ is struck by a unit force impulse $f_n^{p_i}$ which has the inverse direction of the surface normal $n$ at the position. Then, the $k$-th mode amplitude at the position $p_i$ is represented as $a_k^{p_i} = u_k^T f_n^{p_i}$, where $u_k$ is the $k$-th eigenvector.

Using the target frequencies $F$, amplitudes $A$ and simulated parameters, our objective function for minimizing the perceptual difference of the mode frequencies is represented as

$$E_f = \frac{1}{2}\sum_{i=2}^{N}\left(Bark(s_f f_i) - Bark(F_i)\right)^2 \quad (1)$$

where $Bark(f)$ is a function to transform the frequency to critical band rate $[bark]$ [ZF99], and $s_f = F_1'/f_1$ is the scaling factor. The

objective function for amplitudes is also obtained using the balances with other mode amplitudes at the position

$$E_a = \frac{1}{2}\sum_{j=1}^{T}\sum_{i=2}^{N}\left(\frac{a_i^j}{a_{max}^j} - \frac{A_i^j}{A_{max}^j}\right)^2. \quad (2)$$

where $a_{max}^j$ and $A_{max}^j$ denote the largest amplitude at the position $j$ of the simulated and target's modes respectively. These formulations are similar to [BLT*15]; however, we use the perceptual metrics whereas they use square distances of frequencies and amplitudes. We minimize these functions by optimizing the Young's modulus $Y_e \in \mathbb{R}^M$ at each finite element $e$, where $M$ denotes the number of the elements. Finally, our design problem is formulated as

$$\arg\min_{Y_e} : \ w_f E_f + w_a E_a, \ \ subject\ to : Y_e > 0 \quad (3)$$

where $w_f$ and $w_a$ denote the positive weights.

Note that we do not optimize damping parameters. We instead reuse the estimated damping from the assigned sound clips as mode-dependent damping. This means that our damping is not spatially constant. This setting is physically incorrect, but it makes the problem simpler.

## 6. Material Optimization

The optimization of element-wise material parameters is impractical. To reduce the design space of material parameters, we introduce the reduction technique of [XLCB15]. The technique expresses the Young's modulus as $Y = \Phi z$ using the eigenvectors of the volumetric mesh Laplacian $\Phi \in \mathbb{R}^{M\times m}$, and uses the generalized material parameters $z \in \mathbb{R}^m$, $(m << M)$ for the optimization. Then, our design problem can be rewritten in the reduced space as

$$\arg\min_{z} : \ w_f E_f + w_a E_a + w_r R, \ \ R = \frac{1}{2} z^T Q z \quad (4)$$

where $w_r$ is a weight, $R$ is the regularization term, and $Q$ is the reduced Laplacian matrix which is diagonal and its entries consist of the eigenvalues of the volumetric mesh Laplacian (please see [XLCB15] for the details). This material reduction also has a merit to reduce the over-fitting problem.

We solve our design problem Eq. (4) by decomposing it into two problems $min : E_f$ and $min : E_a$, and minimizing them alternately. We employ a hybrid optimization scheme [CLJ09] of evolutional strategies (we used CMA-ES [HMK03]) and gradient descent approach (we employed the Quasi Newton method). For the details of the gradient computation and this hybrid scheme, please see Appendix 2, 3.

## 7. Fast Approximate Modal Analysis

At each iteration during our optimization, a modal analysis is required for the evaluation of the objective function and its gradient. However, standard modal analysis (solving a generalized eigenproblem of large stiffness and mass matrices) is prohibitively expensive and impossible to execute at an interactive rate. To address this, we present a method that combines extended data-driven online coarsening of finite elements (§7.1) and highly parallelized hierarchical component mode synthesis (§7.2).

### 7.1. Data-Driven FEM using Regression Forests

In this section, we explain the data-driven online coarsening of the FEM mesh. It takes the detailed voxel mesh ($2\times2\times2\times$ cube elements) as input and generates a coarse approximated mesh (a cube element) as output using the the material parameter mapping learned from training data in the precomputation step (Figure 4). The concept of our data driven FEM coarsening is based on [CLSM15]. The goal of their data-driven FEM is obtaining

$$(E_1,...,E_8) = DDFEM(e_1,..,e_8) \quad (5)$$

where $DDFEM()$ is a function that takes eight material parameters $(e_1,...,e_8)$ of a detailed mesh and returns the corresponding eight coarse material parameters $(E_1,...,E_8)$ at the cubature points to minimize the error. Their system computes this function for all possible input values in precomputation and stores the result in the main memory. The system then evaluates this function referring the memory at runtime. It aggressively accelerates FEM while maintaining the accuracy by reducing the Dofs $(24/81)$ and the number of the cubature points $(8/64)$ although the total number of the material parameters remains unchanged between the detailed and coarse mesh. However, in their approach, given $N$ discrete materials, the number of material combinations becomes $N^8$. Although they also proposed a compression algorithm by retaining only the small number of representative material combinations, it still cannot be used for our material optimization that requires a large range of continuous material settings. Additionally, it is non-trivial to obtain an actual value from such representative materials. To address this, we present three techniques: 1: Overlapping Free Cell Ordering, 2: Scaling Factor Separation, 3: Regression Forests. The former two techniques reduce the parameter space of the feature vector $e$ (the detailed eight material parameters) for efficient machine learning, and the last technique enables handling of a large amount of dataset with a constant evaluation cost.

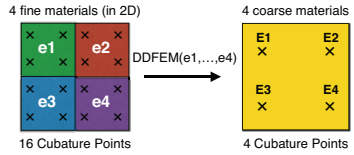


**Figure 4:** *Data-Driven Coarsening [Chen et al. 2015] (in 2D illustration). The function $DDFEM()$ takes four material parameters $(e1,e2,e3,e4)$ of fine four elements (left) and returns corresponding four coarse material parameters $(E1,E2,E3,E4)$ at the quadrature points (right) to minimize the error.*

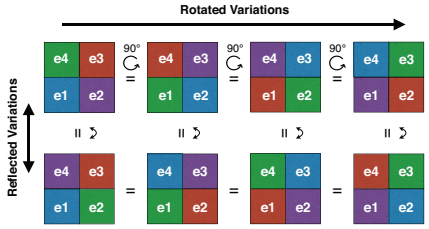


**Figure 5:** *Eight equivalent cell variations (in 2D illustration). The top row represents four rotated variations and the bottom row represents four reflected variations.*

### 7.1.1. Overlapping Free Cell Ordering

As shown in Figure 5, the rotated and reflected variations of a material setting are basically equivalent. To enumerate such patterns increases the parameter space of the feature vector unnecessarily and it should be reduced for efficiency. To address this, we define Overlapping Free Cell Ordering algorithm which makes explicit consideration of rotated and reflected patterns unnecessary.

First, we redefine the data-driven function $DDFEM()$ Eq.5 as.

$$E_i = DDFEM_i(e_1,..,e_8), \ \ i = 1,2,...,8. \quad (6)$$

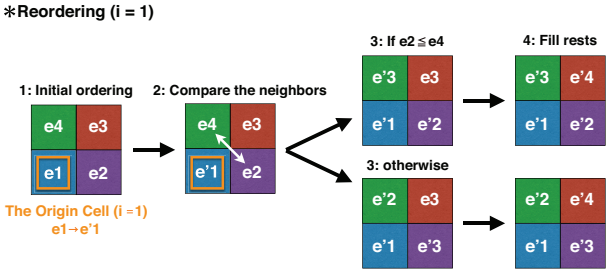


**Figure 6:** *Overlapping Free Cell Ordering (in 2D illustration). 1: At the i-th cell evaluation (in this example, we assume $i = 2$), $e_i$ becomes the origin cell $e'_1$. 2: we compare the material values of the adjacent cells. 3: the smaller cell becomes $e'_2$ and the other becomes $e'_3$. 4: The left cell becomes $e'_4$.*

Our data-driven FEM function returns a scalar while Eq.5 outputs a $\mathbb{R}^8$ vector. It means we repeat this $DDFEM()$ evaluation eight times to convert a detailed $2 \times 2 \times 2$ element into a coarse element. Next, we reorder the numbering of the eight cells by each $DDFEM()$ evaluation. We show this operation as a 2D example in Figure 6. The indices of the cells are defined in a local $\mathbb{R}^3$ space coordinate. At the $i$-th evaluation within the eight evaluations, we define the $i$-th cell as the origin $e'_1$. Then, we compare the value of the Young's modulus of the three adjacent cells of the origin cell (in 2D, two cells), and define the index the cell who has the smallest value as $e'_2$, the cell who has the secondary smallest value as $e'_3$, and the other cell as $e'_4$. Finally, we decide the ordering of the rest four cells by the following rule: The cell that is adjacent to $e'_2$ and $e'_3$ becomes $e'_5$. The cell that is adjacent to $e'_3$ and $e'_4$ becomes $e'_6$. The cell that is adjacent to $e'_2$ and $e'_4$ becomes $e'_7$. The last one becomes $e'_8$.

Then, using these reordered parameters, our $DDFEM()$ function is redefined again as

$$E_i = DDFEM_i(e'_1, e'_2, ..., e'_8), \quad e'_1 = e_i \tag{7}$$

By using this representation, we can avoid explicit enumeration of the eight rotated and eight reflected patterns of a material pattern, and reduce the input parameter space in 3D at both training and runtime. For dataset generation at the training, we first determine the value at the origin cell, and seed the values at the three cells $e'_2$, $e'_3$, $e'_4$ to be $e'_2 \leq e'_3 \leq e'_4$, and the rest of the values are randomly seeded.

### 7.1.2. Scaling Factor Separation

Young's modulus has a large range of the value $10^{-2}$ (Rubber) $\sim 10^3$ (Diamond) *GPa* while Poisson's ratio has a small range $(-1/2,\ 1/2)$. It is difficult to treat a practical amount of data for such a large range during training. To avoid this, we dramatically reduce the training size by separating the scale factor.

Based on [CLSM15], our $DDFEM()$ is constructed to minimize the square difference of the integral of the strain energy density functions between the detailed and coarse meshes.

$$\arg\min_{E_i} \sum_{f \in F} \left( \sum_{i=1}^{8} w_i v_i^c(f, E_i) - \sum_{j=1}^{8} \sum_{i=1}^{8} w_i v_{ji}^d(f, e'_j) \right)^2 \tag{8}$$

where $w$ denotes the cubature weights, $F$ denotes a set of randomly sampled external forces, and $v^c$ and $v^d$ represent the strain energy density function of the coarse and detailed mesh respectively. Here, the strain energy density function in linear elastic is represented as $v(f,e) = K(e)u(e)^2 = K(e)(K^{-1}(e)f)^2$ where $K(e)$ and $f$ are the stiffness matrix and the external forces respectively. In addition, multiplying $e$ by a scalar $s$, $v(f, s \cdot e) = K(s \cdot e)u(s \cdot e)^2 = sK(e)((sK(e))^{-1}f)^2 = v(f,e)/s$ because $K()$ is the linear function of $e$. Then, the minimization problem

$$\arg\min_{E_i} \sum_{f \in F} \left( \sum_{i=1}^{8} w_i v_i^c(f, E_i) - \sum_{j=1}^{8} \sum_{i=1}^{8} w_i v_{ji}^d(f, s \cdot e'_j) \right)^2 \tag{9}$$

is equivalent to

$$\arg\min_{E'_i = E_i/s} \sum_{f \in F} \left( \sum_{i=1}^{8} w_i v_i^c(f, E'_i) - \sum_{j=1}^{8} \sum_{i=1}^{8} w_i v_{ji}^d(f, e'_j) \right)^2 \tag{10}$$

This means that we can separate the input parameter space of our $DDFEM()$ problem by the multiplication of the value of the origin cell as a scale factor and their quotients. Finally, we can obtain our $DDFEM()$ function as

$$E_i = e_i \cdot DDFEM_i\left( \frac{e'_2}{e_i}, ..., \frac{e'_8}{e_i} \right) \tag{11}$$

An advantage of this representation is that it reduces not only the range of dataset but also the dimensions of the feature vector from $\mathbb{R}^8$ to $\mathbb{R}^7$. Note that we assume our model as linear elastics although the original $DDFEM()$ treats nonlinearity because the vibrational analysis discussed in this paper is a linear analysis. Introducing the nonlinearity for large deformation is a future work.

### 7.1.3. Regression Forests

In contract with Chen et al.'s method [CLSM15], we do not construct the database of data-driven materials because of two reasons. First, their database approach cannot handle the inputs that are not included in the training dataset because it has no generalization ability. Second, the evaluation cost at runtime is increased at a rate proportional to the amount of the dataset although the amount of the dataset should be increased for handling more material patterns. To address these problems, we train our $DDFEM()$ function using two regression forests. Our regression forests are similar to [LJS*15] which construct each tree through two steps training: tree structure construction with a subset of learning data and least-square solve for the regression coefficients at each leaf node with all the dataset. The regression forest has an advantage of constant cost evaluation even if the amount of the dataset is increased. Finally, our $DDFEM()$ becomes

$$\begin{cases} E_i = \bar{e} \cdot Reg_1(\frac{e'_2}{\bar{e}}, ..., \frac{e'_8}{\bar{e}}) & (e_i = 0) \\ E_i = e_i \cdot Reg_2(\frac{e'_2}{e_i}, ..., \frac{e'_8}{e_i}) & (e_i > 0) \end{cases} \tag{12}$$

where $Reg()$ represents the regression function, and $\bar{e}$ is the average of the Young's modulus in the target eight cells.

## 7.2. Hierarchical Component Mode Synthesis

After coarsening the mesh, we compute modal analysis using a novel hierarchical component mode synthesis method (HCMS) including an efficient error correction algorithm (Figure 7). It takes the coarse voxel mesh as input and solves a generalized eigenproblem via hierarchical merging. It first decomposes the mesh into small components and solves a generalized eigenproblem for each component. It then hierarchically merges adjacent components and solves generalized eigenproblems for the merged component. Conventional CMS [BC68] computes the eigenmodes of a structure by combining several small local subdomains after decomposing it into several small subdomains. Our HCMS decomposes a structure into finer subdomains compared to conventional CMS to increase the computational efficiency while sacrificing accuracy. To compensate for the loss of accuracy, we apply a subspace iterative error correction using the result of HCMS as an initial solution.

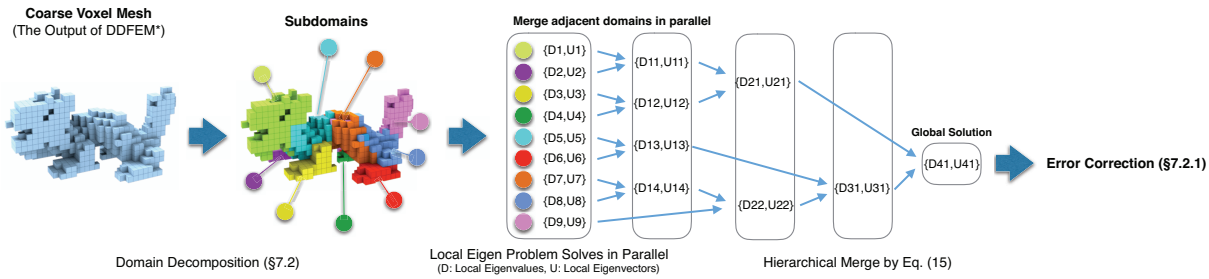


**Figure 7:** *Hierarchical Component Mode Synthesis. After coarsening the mesh, we decompose it into many subdomains and hierarchically merges them with reducing their DoFs in parallel. Finally, we improve the accuracy using an error correction algorithm.*

To simplify the explanation for our HCMS, we first begin with assuming that a model can be decomposed into two non-overlapping domains $S1$ and $S2$ as in conventional CMS, and the eigenpairs of each domain are already known. Under this assumption, the entire stiffness matrix $K_{total}$ and the entire mass matrix $M_{total}$ can be represented as

$$K_{total} = \begin{bmatrix} K_{11} & K_{12} \\ K_{12} & K_{22} \end{bmatrix}, M_{total} = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \quad (13)$$

where $K_{11}, K_{22}$ and $M_1, M_2$ denote the local stiffness and mass matrices of each sub-domain respectively. $K_{12}$ and $K_{21}$ are the interface matrices that connect the domains $S1$ and $S2$. If the eigenvectors of each domain $U_1$ and $U_2$ are already known, we can rewrite the Eq. (13) using the reduced matrices of each domain with remaining the lower frequency modes as

$$K'_{total} = \begin{bmatrix} D_1 & U1^T K_{12} U2 \\ U2^T K_{12}^T U1 & D_2 \end{bmatrix} \quad (14)$$

where $D_1 = U1^T K_{11} U1$ and $D_2 = U2^T K_{22} U2$ are diagonal matrices in which each diagonal entry is the eigenvalue of the respective subdomain. Note that the entire mass matrix also takes the same form for, $U_1^T M_1 U_1 = I$, and $U_2^T M_2 U_2 = I$, meaning that the entire mass matrix becomes an identity matrix. Athough conventional CMS distinguishes the interface of adjacent subdomains and subdomains, and assumes the interface as fixed [BC68] or considers the boundary modes [YXG*13], our approach neither distinguish them nor fix the interface, and does not treat the interface explicitly. We can obtain a reduced eigenproblem of the entire structure as $K'_{total} U'_{total} = \Lambda_{total} U'_{total}$, where $\Lambda_{total}$ is a diagonal matrix in which each diagonal entry is the eigenvalues of the entire domain. We solve this reduced eigenproblem, and finally recover the global eigenvectors by

$$U_{total} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} U'_{total}. \quad (15)$$

We apply the pair wise merger explained above in a hierarchical manner. We divide a large structure into many small subdomains and merge them in a hierarchical manner (Figure 7). The system first decomposes the volumetric mesh after coarsening into many small subdomains $S_1, S_2, ..., S_N$ by a domain decomposition. To decompose a mesh, we use [WLAT14] by expanding it into volumetric mesh, which decomposes the mesh by K-Means++ clustering [AV07] of the eigenvectors of the volumetric mesh Laplacian. It requires no additional precomputation costs since the volumetric mesh Laplacian has been already obtained at the precomputation stage as described in §4.

We compute the local generalized eigenproblem of $N$ subdomains in parallel and reduce the DoFs using the eigenvectors at each subdomain. Next, we iteratively merge two adjacent subdomains by Eq. (14), and solve the reduced eigenproblem, and Eq. (15) to obtain the eigenvectors of the merged subdomain. This procedure also can be executed in parallel until all the subdomains are merged. Finally, we merge all the subdomains and obtain the approximate eigenvector of the entire structure. The order of merging subdomains is irrelevant in our algorithm because the error caused by suboptimal order will be fixed later in our error correction (§7.2.1). We note that this hierarchical merging procedure is new. We implemented local eigenprobrem solves of each subdomain by a combination of incomplete Lanczos matrix triangulation and QR method.

### 7.2.1. Error Correction

HCMS is just an approximation method and sacrifices the accuracy for computational efficiency. To correct this error, we introduce the subspace iteration method [Bat13] using reduced mass Gram-Shmidzt process [YLX*15]. We set approximated eigenvectors of HCMS as the starting iteration vectors $X_0$ and execute the following iteration $k = 1, 2, 3...$ until it converges.

$$Solve\ PCG: \ KU = MX_{k-1} \quad (16)$$
$$U \leftarrow ReducedMGS(U) \quad (17)$$
$$K' = U^T KU, \quad M' = U^T MU \quad (18)$$
$$Solve\ QR: \ K'Q = \Lambda M'Q \quad (19)$$
$$X_k = U + \Sigma(UQ - U) \quad (20)$$
$$X_k \leftarrow ReducedMGS(X_k) \quad (21)$$

where $ReducedMGS()$ is the reduced mass Modified Gram-Schmidt process to orthogonalize the eigenvectors [YLX*15], $\Sigma$ denotes a diagonal matrix in which each diagonal corresponds to the overrelaxation weight of the $i$-th eigenvalue to accelerate the convergence [BR80]. We solve the first line Eq. (16) by incomplete cholesky factorized pre-conditioned conjugate gradient method with respect to each column vector in parallel, and implement the QR method Eq. (19) on GPU.

## 8. Results

### 8.1. Validation of Modal Analysis

In this subsection, we verify the accuracy and computational efficiency of our fast approximate modal analysis. As the ground truth, we used the result of the full-DoF standard modal analysis using ARPack (with sufficiently fine-resolution uniform hexahedral mesh). We used CPU: Intel Core i7 2.6 GHz, RAM: 16GB, GPU: NVIDIA GeForce GT 750M as the equipments in §8 excluding the DDFEM trainings. We set the Poisson's ratio as 0.25 and the density as 1.0 $kg/m^3$ for all experiments.

**Data-Driven FEM:** We call our data-driven FEM as extended data-driven FEM (DDFEM*) for distinguishing from Chen et al.'s method [CLSM15] (DDFEM). We used two regression forests, and

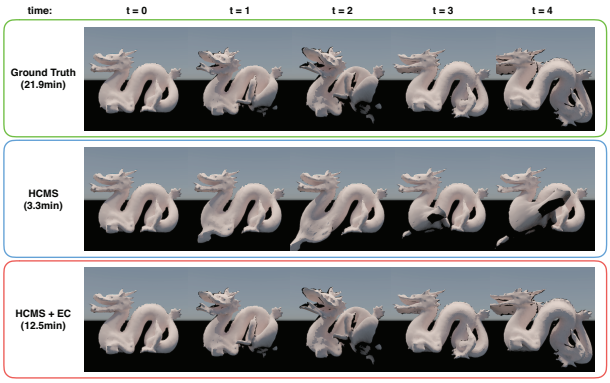


**Figure 8:** *Comparison of the deformation of the 7-th modes between HCSM with/without EC and the ground truth. Our error correction algorithm efficiently improve the accuracy within an additional few minutes.*

| Method | Training Time | Evaluation Cost | Storage | Error |
|---|---|---|---|---|
| Native Coarsening | zero | 1 µs | zero | 0.0383003 |
| Chen et al. 2015 | 2 hours | 1 ms | 3.82 MB | 3.88114E-05 |
| Ours (DDFEM*) | 12.5 days | 0.2 ms | 529.9 MB | 4.19069E-05 |

With [0, 10] GPa Young's modulus
(trained range)

| Method | Training Time | Evaluation Cost | Storage | Error |
|---|---|---|---|---|
| Native Coarsening | zero | 1 µs | zero | 0.00360124 |
| Chen et al. 2015 | 2 hours | 2 .3 ms | 8.11 MB | 3.66629E-05 |
| Ours (DDFEM*) | zero | 0.2 ms | 529.9 MB | 3.6882E-05 |

With [100, 10000] GPa Young's modulus
(outrange of the trained dataset for our regression forests)

**Figure 9:** *Comparison of the accuracy of data-driven FEM. Top: The results with [0, 10] GPa range of Young's modulus (trained range of our regression forests). Bottom: The results with [100, 10000] GPa range of Young's modulus (untrained range of our regression forests).*

three regression trees for each forest, and set the maximum depth of all the trees as 20. We trained each regression forest for DDFEM* by 1 billion entries of the dataset for constructing the tree structures and 10 billion entries of the dataset for training each leaf node (regression function construction). For the dataset generation of data-driven FEM (a sample includes 8 material parameters of detailed $2 \times 2 \times 2$ blocks and the corresponding 8 coarse material parameters), we used 1,000 force directions and sample 5 sample magnitudes in each direction, resulting in 5,000 force samples for each material combination. We seeded the material combinations randomly with [0, 10] *GPa* of range of Young's modulus using hypercube sampling. The training time took about ten days for data generation, two days for the tree structure training using clusters of 12 computers, and a half day for training the leaves. Finally, our regression forests required 529.9 MB for storage.

We verified the accuracy and the evaluation cost of our data-driven FEM by comparing it with a native coarsening approach (simply averaged material setting of the detailed elements) and Chen et al's method [CLSM15]. We prepared 10,000 detailed $2 \times 2 \times 2$ FEM cube blocks with randomized material distribution and applied them to 2,000 random external force samples. We define the error of coarsening as the average of the square distances of the displacements between detailed simulation (ground truth) and coarse simulation over the samples. Because the detailed elements and the coarse element have different numbers of vertices, we measured the error by creating a detailed mesh from the coarse simulation by trilinear interpolation and computing the distance of their displacements. In addition, we defined the evaluation cost as the time for coarsening a $2 \times 2 \times 2$ elements to an element. Since the evaluation cost of DDFEM depends on the database size and the search algorithm, we then prepared the sorted index of the database at precomputation, and searched them by quick search at runtime.

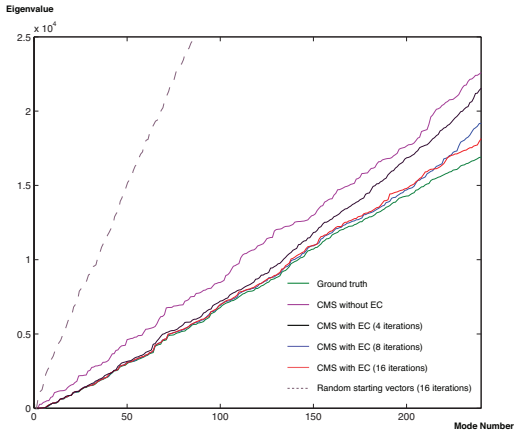


**Figure 10:** *The accuracy of HCMS with/without EC. Our error correction algorithm dramatically improves the accuracy within a few iterations. The horizontal axis: the mode number, the vertical axis: the eigenvalues.*

Figure 9:Top shows the result of this experiment for the samples with randomly generated Young's modulus setting using [0,10] *GPa* range. This material parameter range is included in the training dataset for our regression forests. Our regression forests successfully reduce the error on a level with Chen et al.'s method [CLSM15] while native coarsening approach causes a large error. Next, Figure 9:Bottom shows the result by the samples with the range of Young's modulus [100,10000] *GPa* that is clearly out of range of the trained dataset for DDFEM*. The result shows our method can also handle this range of inputs and returns good results with no additional training while DDFEM requires additional trainings for new data. It is enabled by our scaling parameter separation algorithm. In addition, the evaluation cost of our DDFEM* is constant even if the amount of dataset is increased and much faster than DDFEM whose cost is increased in proportion to the amount of the dataset.

**Hierarchical Component Mode Synthesis:** For the evaluation of HCMS, we decomposed each model of Figure 11 into 10∼20 subdomains and hierarchically merged them. When two subdomains are merged, we retained $min(N_{sub}, 512)$ DoFs where $N_{sub}$ denotes the total DoFs of the two subdomains.

Figure 10 shows the error comparison of eigenvalue computation of HCMS with/without error correction (EC) and the ground truth using the Chinese dragon model. We can see that our error correction algorithm converges very quickly in only a few iterations and efficiently reduces the error. In addition, the break line in Figure 10 shows comparison of the error correction using the result of HCMS and randomized (with $N(0,1)$ Gaussian) and orthogonalized vectors as the starting vectors. We can see that the iteration with randomized starting vectors does not converge within a few iterations. This shows that the approximate solution of HCMS is a good starting vectors for the subspace iteration method.

Next, we show the comparisons between the modal deformations by standard modal derivatives and our method's (HCMS and HCMS + EC) at the 7th mode in Figure 8 for example. The results show that our HCMS can capture rough motions of the elastic

| Model | DoF | Precomputation | ARPack | DDFEM* | HCMS | HCMS + EC | DDFEM*+ HCMS + EC |
|---|---|---|---|---|---|---|---|
| Stanford Bunny | 31419 | 18.8m | 4.1h | 15.3m | 9.8m | 28.4m | 2.6s |
| Asian Dragon | 6009 | 23s | 14.8m | 2.1m | 1.1m | 4.3m | 0.86s |
| Utah Teapot | 12057 | 1.5m | 52.8m | 5.7m | 4.9m | 17.2m | 1.1s |
| Chinese Dragon | 11394 | 50s | 21.9m | 3.2m | 3.3m | 12.5m | 1.4s |
| Pitcher | 15927 | 2.3m | 37,6m | 6.4m | 3.4m | 17.8m | 2.3s |
| Snare Drum | 35484 | 21.5m | 3.9h | 12.3m | 9.7m | 25.6m | 2.4s |

**Figure 11:** *Computation time comparison of modal analysis. We computed the first 256 modes for all model.*

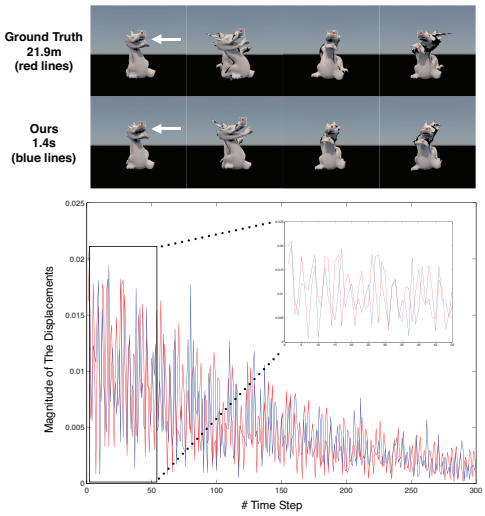


**Figure 12:** *Comparison of between two deformation trajectories of the dragon's nose (red circle at the top thumbnails) produced with standard modal derivatives (red) and our method (DDFEM* + HCMS + EC) (blue). The white arrows at the top thumbnails represent the applied force impulse to drive them.*

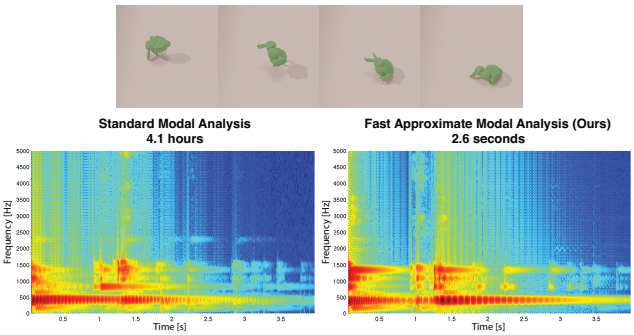


**Figure 13:** *Comparison of the modal sound synthesis from a simple rigid body physics simulation between standard modal analysis (left) and our fast approximate modal analysis (right).*

object in both lower and higher frequency domains, and the error correction algorithm successfully brings the approximate solution close to to the ground truth within an additional few minutes.

**Combination of Extended Data-Driven FEM and HCMS:** Figure 11 shows the computational times of each modal analysis method using DDFEM*, HCMS with/without EC, and their combination with several models, respectively, while ARPack denotes the standard modal analysis (conventional method). The precomputation column in Figure 11 represents the precomputation times taken for each model (The voxelization and the eigenproblem solves for the volumetric Laplacian matrix). The computation times of DDFEM* include the time for the online coarsening procedure. We achieve two orders of magnitudes acceleration with each of DDFEM* and HCMS + EC respectively, and three orders of magnitudes acceleration in total compared to the conventional modal analysis by their combination. The exact computational time using the combination method becomes 0.5∼3.0 secs, which is acceptable for interactive evaluation.

Figure 12 shows comparison between two deformation trajectories produced with standard modal derivatives (ground truth) and our fast approximate modal analysis method after applying a unit force impulse at the location pointed by the white arrow in the top thumbnails. We also provide the time series of magnitudes of the displacement at the dragon's nose in Figure 12:Bottom. The two trajectories plot quite a similar form, which shows that our method gives good approximation for the conventional approach with much faster operation.

Finally, Figure 13 shows the spectrograms of the sound produced by a rigid body physics animation using ground truth and our combination method. Naturally, the spectrogram of the ground truth includes more high frequency components than that of ours because it uses a detailed FEM mesh. However, our result can capture a better portion of the major components enough for our optimization problem.

### 8.2. Physically-Based Sound Design

In this subsection, we demonstrate our physically-based sound design framework. For all the examples, we used the results of our fast approximate modal analysis to render the sound. We set the weights in Eq. (4) to $w_f = 1.0$, $w_a = 10.0$, $w_r = 10^{-5}$ in our experiment.

**Basic Sound Assignment:** Figure 14 shows an example of assigning two sounds to a frying pan model. The frying pan consists of a handle made of wood and plate made of iron. We stuck the pan at the handle and the plate, and recorded the respective sounds. We used these recorded sounds as input to the system. In this example, 30 extracted target modes were extracted from these two sound clips, and we controlled the first 30 modes of the model excluding the six rigid modes. The two spectrograms at the top row in Figure 14 represent the rendered sounds when each position is struck before the optimization. The spectrograms at the middle row are target sounds, and at the bottom row are the results after one minute of optimization. Apparently, the two spectrograms after the optimization closely resemble each target sound. In addition, even if a different position from the one assigned is struck, the sound characteristics of the target sounds near the position is produced in a physically plausible manner (Figure 14:Bottom). This result shows that over-fitting is not a serious problem in our optimization. Furthermore, our approach requires less amount of example sounds for designing the sound of an object, which reduces the user's effort. For example, in [vdDKP01], there is a frying pan example which is similar to our experiment. They used five example sounds to design the sound of the plate alone (except the handle) while we used only one example sound for each part.

**Interactive Editing:** Next, we demonstrate an example of the interactive editing procedure of physically-based sound using a

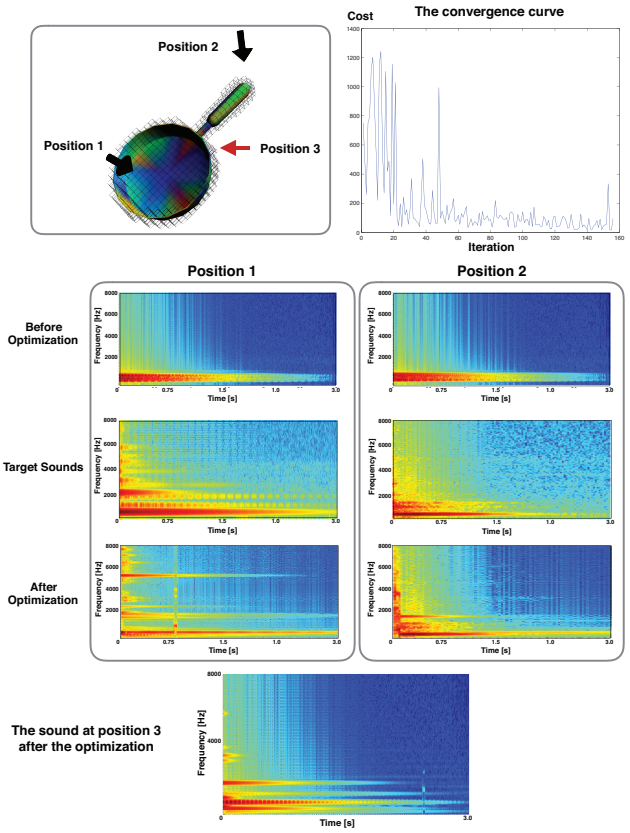


**Figure 14:** *The result of assigning two target sounds to a frying pan model after a minute of optimization. We assigned the metal sound to position 1 (plate) and wooden sound to position 2 (handle). Top right shows the convergence curve of the cost (Eq. 4)*

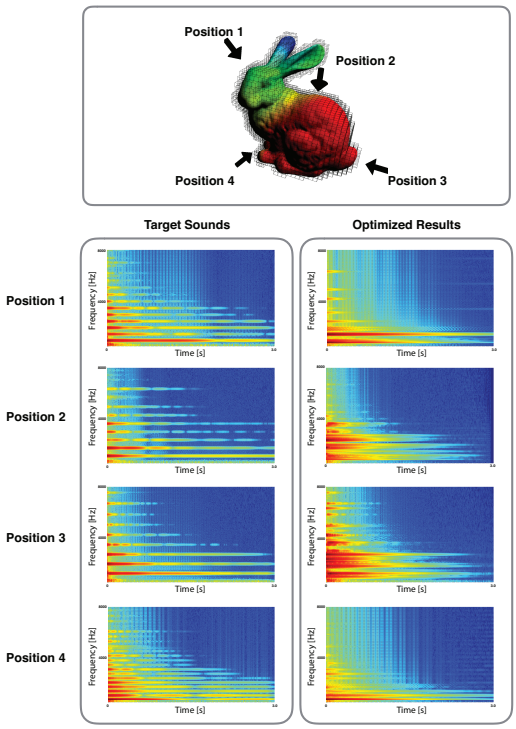


**Figure 15:** *The result of assigning four target sounds to stanford bunny model after four minutes of optimization.*

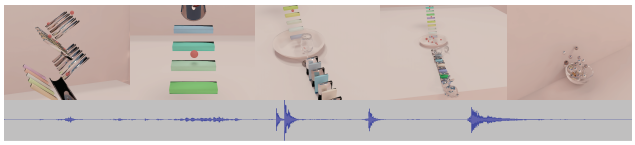

**Figure 16:** *The result animation including various objects.*

teapot model. Please see the supplemental video for an interactive demonstration. We first assigned a sound caused by a metal plate being hit to the teapot's body and started the optimization. During the optimization, we checked the intermediate result on the UI view (Figure 3) and stopped the iteration when the sound of the object was sufficiently close to the given target sound. In this timing, all the positions indicated sounds similar to the target sound. To append more varied sound properties, we assigned two additional target sounds (two different metal sounds) to the lid and spout one by one. As seen in this example, the user can design the sound of an object while running the optimization and the user's edits are immediately reflected in the simulation within a few seconds. The user can iteratively re-edit the sound property of the object with checking the intermediate results. This type of interactive physically-based sound design workflow has not been presented before.

**Imaginary Sound Assignment:** Our system allows the user to assign imaginary target sounds to a 3D model (Figure 15). In this example, we assigned a piano C4 sound to the head of the bunny, a piano E4 sound to the body, a piano G4 sound to the tail, and a piano F3 sound to the leg, and optimized. The result after four minutes of optimization is shown in Figure 15 by spectrograms. Although an object that has such sounds does not exist in the real world, our system produces a physically convincing result. This is also an advantage of our approach compared to the previous method [RYL13].

**A Complicated Scenario Example including Various Objects:** Finally, we demonstrate a complicated scenario involving several objects as shown in Figure 16. We include this animation scene in the accompanying video. In this scene, the sounds of the all objects are designed by our system, and all sounds are triggered automatically by rigid body simulations. It took 30 minutes for us to design sound of all the object in the scene using our tool, one week for rigid-body simulator setting, and three days for visual rendering.

## 9. Conclusion and Future Work

We presented a novel example-based design framework of physically-based sound of a 3D model using material optimization. In addition, our material optimization is third orders of magnitudes accelerated to the level of interactive rate by a novel fast approximate modal analysis that consists of data-driven online coarsening of the mesh and hierarchical component mode synthesis with efficient error correction. We demonstrated our framework provides the user to intuitive design workflow of the sound of an object with a set of examples.

However, several limitations are observed in our work, which remain to be addressed in future work. A critical limitation is that our sound design technique cannot be used for fabrication because we employ continuous material optimization, which simulates materials that do not exist in the real world. Although the usage of combinatorial optimization using existing materials can also be considered, it is still difficult to make an object consist of various materials that have a large range of material parameters seamlessly using existing fabrication tools. Second, our current optimization does not

consider the effects of sound radiation and propagation. However, radiation and propagation effects are important for sound design and thus we plan to extend our method to support them using Li et al. [LFZ15]'s technique in the future.

We only optimized Young's moduli in this paper. which limits the reproducibility of the example sounds. As an improvement, Poisson's ratios could be treated similarly by converting Young's moduli and Poisson's ratios to the 2-dimensional space of Lame' parameters [STC*13] and performing the optimization in this linear space. However, the dimension of the feature vector for inputting data-driven FEM increases, and it could reduce the training efficiency of our regression forests. There is a similar problem for treating the densities. To treat such a large parameter space with a machine learning technique remains as future work.

Naturally, our fast approximate modal analysis pipeline can also be used for deformable animation. Applying our approach to large deformable simulation or combining it with [YLX*15] could be also useful and promising work. Finally, we use voxel elements. The fineness of the details of the model depends on the voxel resolution. To address this, adaptive coarsening could be useful. However, how to treat such an adaptive mesh with machine learning technique is non-trivial and remains as future work.

## References


[Adr91] ADRIEN A.-M.: *The missing link: Modal synthesis. In Representations of Musical Signals*. MIT Press Cambridge, 1991. 1

[AV07] ARTHUR D., VASSILVITSKII S.: k-means++: The advantages of careful seeding. *In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematic* (2007), 1027–1024. 6

[Bat13] BATHE K.-J.: The subspace iteration method - revisited. *Computers and Structures 126* (2013), 977–983. 3, 6

[BC68] BAMPTOM M. C. C., CRAIG R. R.: Coupling of substructures for dynamic analyses. *AIAA Journal 6*, 7 (1968), 1313–1319. 1, 2, 5, 6

[BLT*15] BHARAJ G., LEVIN D. I. W., TOMPKIN J., FEI Y., PFISTER H., MATUSIK W., ZHENG C.: Computational design of metallophone contact sounds. In *ACM Trans. on Graphics (SIGGRAPH Asia 2015)* (2015), vol. 34, pp. 223:1–223:13. 2, 4

[BR80] BATHE K. J., RAMASWAMY S.: An accelerated subspace iteration method. *Computer Methods in Applied Mechanics and Engineering 23* (1980), 313–331. 6

[CLJ09] CHEN X., LIU X., JIA Y.: Combining evolution strategy and gradient descent method for discriminative learning of bayesian classifiers. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (2009), no. 8, pp. 507–514. 4

[CLSM15] CHEN D., LEVIN D. I. W., SUEDA S., MATUSIK W.: Data-driven finite elements for geometry and material design. In *ACM Trans. on Graphics (SIGGRAPH 2015)* (2015), vol. 34, pp. 74:1–74:10. 1, 2, 4, 5, 7

[CP88] CHEN S.-H., PAN H. H.: Guyan reduction. *Communications in Applied Numerical Methods 4*, 4 (1988), 549–556. 2

[CvdDLH07] CORBETT R., VAN DEN DOEL K., LLOYD J. E., HEIDRICH W.: Timbrefields: 3d interactive sound models for real-time audio. *Presence: Teleoperators and Virtual Environments - Special section: Advances in interactive multimodal telepresent systems 16*, 6 (2007), 643–654. 2

[HMK03] HANSEN N., MULLER S., KOUMOUTSAKOS P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). In *Evolutionary Computation* (2003), no. 11, pp. 1–18. 4

[HSO03] HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformation using modal analysis with constraints. In *In Proc. of Graphics Interface* (2003), pp. 247–256. 2

[Hur65] HURTY W. C.: Dynamic analysis of structural systems using component modes. *AIAA Journal 3* (1965), 678–685. 2

[LFZ15] LI D., FEI Y., ZHENG C.: Interactive acoustic transfer approximation for modal sound. *ACM Transactions on Graphics (TOG) 35*, 1 (2015). 10

[LJS*15] LADICKÝ L., JEONG S., SOLENTHALER B., POLLEFEYS M., GROSS M.: Data-driven fluid simulations using regression forests. In *ACM Trans. on Graphics (SIGGRAPH Asia 2015)* (2015), vol. 34, pp. 199:1–199:9. 5

[LRG11] LLOYD D. B., RAGHUVANSHI N., GOVINDARAJU N. K.: Sound synthesis for impact sounds in video games. In *In Symposium on Interactive 3D Graphics and Game* (2011), ACM, p. 7. 2

[MB07] MAXWELL C. B., BINDEL D.: Modal parameter tracking for shape-changing geometric objects. In *Proceedings of the 10th International Conference on Digital Audio Effects (DAFx 07)* (2007). 2

[OCE01] O'BRIEN J. F., COOK P. R., ESSL G.: Synthesizing sounds from physically based motion. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), pp. 529–536. 1

[PvdDJ*01] PAI D. K., VAN DEN DOEL K., JAMES D. L., LANG J., LLOYD J. E., RICHMOND J. L., YAU S. H.: Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), pp. 87–96. 2

[RYL13] REN Z., YEH H., LIN M. C.: Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics (TOG) 32*, 1 (2013), 987–995. 2, 3, 9

[STC*13] SKOURAS M., THOMASZEWSKI B., COROS S., BICKEL B., GROSS M.: Computational design of actuated deformable characters. In *ACM Trans. on Graphics (SIGGRAPH 2013)* (2013), vol. 32, pp. 82:1–82:10. 10

[UMIT10] UMETANI N., MITANI J., IGARASHI T., TAKAYAMA K.: Designing custom-made metallophone with concurrent eigenanalysis. In *In Proceedings of the Conference on New Interfaces for Musical Expression (NIME)* (2010), ACM, pp. 26–20. 2

[vdDKP01] VAN DEN DOEL K., KRY P. G., PAI D. K.: Foleyautomatic: physically-based sound effects for interactive simulation and animation. *In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2001)* (2001), 537–544. 2, 8

[WLAT14] WANG H., LU T., AU O. K.-C., TAI C.-L.: Spectral 3d mesh segmentation with a novel single segmentation field. *Graphical Models 76*, 5 (2014), 440–456. 6

[XLCB15] XU H., LI Y., CHEN Y., BARBIČ J.: Interactive material design using model reduction. *ACM Transactions on Graphics (TOG) 34*, 2 (2015), 14. 3, 4

[YJK13] YUA Y., JANG I. G., KWAKA B. M.: Topology optimization for a frequency response and its application to a violin bridge. *Journal Structural and Multidisciplinary Optimization 48* (2013), 4627–636. 2

[YJKK10] YUA Y., JANG I. G., KIMC I. K., KWAKA B. M.: Nodal line optimization and its application to violin top plate design. *Journal of Sound and Vibration 329* (2010), 4785–4796. 2

[YLX*15] YANG Y., LI D., XU W., TIAN Y., ZHENG C.: Expediting precomputation for reduced deformable simulation. In *ACM Trans. on Graphics (SIGGRAPH Asia 2015)* (2015), vol. 34, pp. 243:1–243:13. 3, 6, 10

[YNY*10] YAMASAKI S., NISHIWAKI S., YAMADA T., IZUI K., YOSHIMURA M.: A structural optimization method based on the level set method using a new geometry-based re-initialization scheme. *International Journal for Numerical Methods in Engineering 12* (2010), 1580–1624. 2

[YVC13] YINA J., VOSSB H., CHENA P.: Improving eigenpairs of automated multilevel substructuring with subspace iterations. *Computers and Structures 119* (2013), 115–124. 2

[YXG*13] YANG Y., XU W., GUO X., ZHOU K., GUO B.: Boundary-aware multidomain subspace deformation. In *IEEE Transactions on Visualization and Computer Graphics* (2013), vol. 19, pp. 1633–1645. 6

[ZF99] ZWICKER E., FASTL H.: *Psychoacoustics: Facts and models, 2nd updated edition*. Vol. 254. Springer New York, 1999. 3

[ZJ11] ZHENG C., JAMES D. L.: Toward high-quality modal contact sound. In *ACM Trans. on Graphics (SIGGRAPH 2011)* (2011), vol. 30, p. 38. 1