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

Kazuhiko Yamamoto1  Takeo Igarashi1
1The 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.

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 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.
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 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-Bampton 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
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].

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-like 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 (f1,...,fN), corresponding dampings (D1,...,DN), corresponding residuals (R1,...,RN), and corresponding amplitudes at T sample positions (A1,....,AN). We call these extracted parameters as target parameters.

For a given finite element mesh, we compute the first N mode frequencies (f1,...,fN) and corresponding amplitudes at T sample positions (a1,....,aN) using modal analysis. The modal analysis computes a generalized eigenproblem: \[ KU = AU, \]

where K and M denote the stiffness and mass matrix respectively and A and U denote the eigenvalues and the corresponding eigenvectors. To compute the mode amplitudes, we assume each sample position p_i (i = 1,...,N) is struck by a unit force impulse f_{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 \[ a_k = u_k^T f_{p_i}, \]

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

\[ E_f = \frac{1}{2} \sum_{i=2}^{N} (\text{Bark}(s_f f_i) - \text{Bark}(F_i))^2 \]

where \( \text{Bark}(f) \) is a function to transform the frequency to critical band rate [bark] [ZF99], and \( s_f = F_i / f_1 \) is the scaling factor. The

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

$$E_a = \frac{1}{2} \sum_{j=1}^{N} \left( \frac{a_j}{a_{\text{max}}} - \frac{A_j}{A_{\text{max}}} \right)^2,$$

(2)

where $a_{\text{max}}$ and $A_{\text{max}}$ 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. Finally, our design problem is formulated as

$$\arg \min_{\mathbf{z}} \: w_f E_f + w_a E_a, \quad \text{subject to} \: \mathbf{Y} e > 0,$$

(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 $Y = \Phi_{E1}$ using the eigenvectors of the volumetric mesh Laplacian $\Phi \in \mathbb{R}^{M \times N}$ and uses the generalized material parameters $\mathbf{z} \in \mathbb{R}^{N}$, ($m \ll M$) for the optimization. Then, our design problem can be rewritten in the reduced space as

$$\arg \min_{\mathbf{z}} \: w_f E_f + w_a E_a + w_i R, \quad R = \frac{1}{2} Q \mathbf{z},$$

(4)

where $w_i$ 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 $\arg \min : E_f$ and $\arg \min : E_a$, and minimizing them alternately. We employ a hybrid optimization scheme [CLJ09] of evolutionary 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 ($\S 7.1$) and highly parallelized hierarchical component mode synthesis ($\S 7.2$).

7.1. Data-Driven FEM using Regression Forests

In this section, we explain the data-driven approach to coarsening of the FEM mesh. It takes the detailed voxel mesh ($2 \times 2 \times 2$ 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

$$(E_1, \ldots, E_8) = DDFEM(e_1, \ldots, e_8),$$

(5)

where $DDFEM(\cdot)$ is a function that takes eight material parameters $(e_1, \ldots, e_8)$ of a detailed mesh and returns the corresponding eight coarse material parameters $(E_1, \ldots, E_8)$ 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.

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 $DDFEM(\cdot)$ Eq. 5 as

$$E_i = DDFEM_i[e_1, \ldots, e_8], \quad i = 1, 2, \ldots, 8.$$
Reordering (i = 1)

1: Initial ordering

2: Compare the neighbors

3: If $e_2 \leq e_4$

4: Fill rests

The Origin Cell (i = 1)

$e_1 \rightarrow e'_1$

$e_1 \rightarrow e_2$

$e_1 \rightarrow e_4$

$e_1 \rightarrow e_3$

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 2 x 2 x 2 x 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 cell $e_i^t$. 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 $e_{i}^t$, the cell who has the secondary smallest value as $e_{i}'$, and the other cell as $e_{i}''$. Finally, we decide the ordering of the rest four cells by the following rule: The cell that is adjacent to $e_{i}'$ and $e_{i}''$ becomes $e_{i}''$. The cell that is adjacent to $e_{i}'$ and $e_{i}''$ becomes $e_{i}'$. The cell that is adjacent to $e_{i}''$ becomes $e_{i}''$. The last one becomes $e_{i}''$.

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

$$E_i = DDFEM_1(e_1', e_2', ..., e_8')$$

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 $e_1'$, $e_2'$, $e_3'$ to be $e_2' \leq e_3' \leq e_1'$, 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) ~ $10^9$ (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.

$$\arg\min_{E_i} \sum_{f \in F} \left( \frac{8}{j=1} \sum_{j=1}^{8} w_{f}(j, E_i) - \frac{8}{j=1} \sum_{j=1}^{8} w_{f}(j, E_i) \right)^2$$

where $w$ denotes the cubature weights, $F$ denotes a set of randomly sampled external forces, and $v'$ and $v''$ 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 $v(f, e) = K(e)u(f, e)^2 = K(e)(K^{-1}(e)f)^2$ where $K(e)$ and $f$ are the stiffness matrix and the external forces respectively. In addition, multiplying $e$ by a scalar $s$, $v(s, f, e) = K(s \cdot e)u(s \cdot e)^2 = sK(e)(sK(e)^{-1} f)^2 = v(f, e)/s$ because $K()$ is the linear function of $e$. Then, the minimization problem

$$\arg\min_{E_i} \sum_{f \in F} \left( \frac{8}{j=1} \sum_{j=1}^{8} w_{f}(j, E_i) - \frac{8}{j=1} \sum_{j=1}^{8} w_{f}(j, E_i) \right)^2$$

is equivalent to

$$\arg\min_{E_i} \sum_{f \in F} \left( \frac{8}{j=1} \sum_{j=1}^{8} w_{f}(j, E_i) - \frac{8}{j=1} \sum_{j=1}^{8} w_{f}(j, E_i) \right)^2$$

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

$$E_i = e_i \cdot DDFEM\left( \frac{e_2}{e_1}, \frac{e_3}{e_1}, ..., \frac{e_8}{e_1} \right)$$

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 $\mathbb{R}^2$. 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

$$\begin{align*}
E_i &= \hat{e} \cdot \text{Reg} \left( \frac{e_2}{e_1}, ..., \frac{e_8}{e_1} \right) & (e_i = 0) \\
E_i &= e_i \cdot \text{Reg} \left( \frac{e_2}{e_1}, ..., \frac{e_8}{e_1} \right) & (e_i > 0)
\end{align*}$$

where $\text{Reg}()$ represents the regression function, and $\hat{e}$ 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.
To simplify the explanation for our HCMS, we first begin with assuming that a model can be decomposed into two non-overlapping domains $S_1$ and $S_2$ as in conventional CMS, and the eigenpairs of each domain are already known. Under this assumption, the entire stiffness matrix $K_{\text{total}}$ and the entire mass matrix $M_{\text{total}}$ can be represented as

$$ K_{\text{total}} = \begin{bmatrix} K_{11} & K_{12} \\ K_{12} & K_{22} \end{bmatrix}, \quad M_{\text{total}} = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} $$ (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 $S_1$ and $S_2$. 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

$$ K'_{\text{total}} = \begin{bmatrix} D_1 & U_1 \end{bmatrix} K_{\text{total}} \begin{bmatrix} U_1^T & D_2 \end{bmatrix} $$ (14)

where $D_1 = U_1^T K_{11} U_1$ and $D_2 = U_2^T K_{22} U_2$ 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. Although 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

$$ K'_{\text{total}} U'_{\text{total}} = \lambda_{\text{total}} U'_{\text{total}}, $$

where $\lambda_{\text{total}}$ 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

$$ U_{\text{total}} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} U'_{\text{total}}. $$ (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 [WLA+14] 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 eigenproblem 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-Schmidt 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.

$$ \text{Solve PCG: } KU = MU $$ (16)

$$ U \leftarrow \text{ReducedMGS}(U) $$ (17)

$$ K' = U^T KU, \quad M' = U^T MU $$ (18)

$$ \text{Solve QR: } K'Q = \Lambda M'Q $$ (19)

$$ X_k = U + \Sigma(UQ - U) $$ (20)

$$ X_k \leftarrow \text{ReducedMGS}(X_k) $$ (21)

where $\text{ReducedMGS()}$ 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...
Comparison of the deformation of the 7-th modes between HCMS with/without EC and the ground truth. Our error correction algorithm efficiently improve the accuracy within an additional few minutes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Time</th>
<th>Evaluation Cost</th>
<th>Storage</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native Coarsening</td>
<td>zero</td>
<td>1 μs</td>
<td>zero</td>
<td>0.0386003</td>
</tr>
<tr>
<td>Chen et al. 2015</td>
<td>2 hours</td>
<td>1 ms</td>
<td>3.82 MB</td>
<td>3.8811E-05</td>
</tr>
<tr>
<td>Ours (DDFEM*)</td>
<td>12.5 days</td>
<td>0.2 ms</td>
<td>529.9 MB</td>
<td>4.19069E-05</td>
</tr>
<tr>
<td>Ours (DDFEM*)</td>
<td>zero</td>
<td>0.2 ms</td>
<td>529.9 MB</td>
<td>3.6882E-05</td>
</tr>
</tbody>
</table>

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

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 x 2 x 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 range of Young’s modulus (trained range of our regression forests).

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 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.

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Time</th>
<th>Evaluation Cost</th>
<th>Storage</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native Coarsening</td>
<td>zero</td>
<td>1 μs</td>
<td>zero</td>
<td>0.0386003</td>
</tr>
<tr>
<td>Chen et al. 2015</td>
<td>2 hours</td>
<td>1 ms</td>
<td>3.82 MB</td>
<td>3.8811E-05</td>
</tr>
<tr>
<td>Ours (DDFEM*)</td>
<td>12.5 days</td>
<td>0.2 ms</td>
<td>529.9 MB</td>
<td>4.19069E-05</td>
</tr>
<tr>
<td>Ours (DDFEM*)</td>
<td>zero</td>
<td>0.2 ms</td>
<td>529.9 MB</td>
<td>3.6882E-05</td>
</tr>
</tbody>
</table>

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

With [100, 10000] GPa Young’s modulus (out-range of the trained dataset for our regression forests)
Computation time comparison of modal analysis. We computed the first 256 modes for all models.

Figure 11: Computation time comparison of modal analysis.

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 pre-computation column in Figure 11 represents the pre-computation times taken for each model. The white arrows at the top thumbnails represent the applied force impulse to drive them.

The white arrows at the top thumbnails produced with standard modal derivatives (ground truth) and our method (DDFEM* + HCMS + EC) (blue). The white arrows at the top thumbnails produced with standard modal derivatives (ground truth) and our method (DDFEM* + HCMS + EC) (blue) The white arrows at the top thumbnails represent the applied force impulse to drive them.

Figure 12: Comparison of 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 12: Comparison of 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 the ground truth within an additional few minutes.

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 $w_f = 1.0, w_d = 10.0, w_v = 10^{-5}$ 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 of Figure 14 represent the rendered sounds when each position is struck after the 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 [vDGRP01], 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...
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 Lamé’ 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 or combination 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
[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
[HMK03] HANSEN N., MULLER S., KOUMOUTSAKOS P.: Reducing the time complexity of the dimensioned evolution strategy with covariance matrix adaptation (cma-es). In Evolutionary Computation (2003), no. 11, pp. 1–18. 4