

VibraVerse: A Large-Scale Geometry-Acoustics Alignment Dataset for Physically-Consistent Multimodal Learning

Supplementary Material

1. Overview

In this supplemental material, we first present the underlying principles of modal analysis and sound synthesis for physically based simulation Sec. 2. Next, we provide additional implementation details for each benchmark task, along with a comprehensive description of the experimental setup Sec. 3. Finally, we introduce a new task not included in the main text, **sound-guided solid identification**, which has not been explored in prior work Sec. 4. This new task further demonstrates the versatility and strength of our dataset.

2. Modal Analysis and Sound Synthesis Details

2.1. Matrix Assembly from Tetrahedral Meshes

The process of deriving the global mass (M) and stiffness (K) matrices from a volumetric tetrahedral mesh is a foundational step in the Finite Element Method (FEM) [11]. We begin with a 3D object discretized into a set of E tetrahedral elements. The entire mesh consists of n nodes (vertices). Each node possesses three translational degrees of freedom (DOFs)—in x , y , and z directions—resulting in a total of $N_{dof} = 3n$ DOFs for the entire system. Both M and K will be $N_{dof} \times N_{dof}$ matrices. The global matrices are constructed by summing the contributions from all individual element matrices (m^e and k^e). We implemented the assembly of matrices M and K based on the code provided in DiffSound [8].

2.1.1. Element-level Matrix Formulation

For each tetrahedron element e in the mesh, we compute its local mass and stiffness matrices. A standard linear tetrahedron has 4 nodes, and thus $4 \times 3 = 12$ local DOFs. Its local matrices m^e and k^e are therefore 12×12 .

Element Mass Matrix (m^e). The element mass matrix m^e represents the distribution of kinetic energy within the element. It is computed from the material’s density ρ and the element’s shape functions N_i . The displacement u at any point within the element is interpolated from the element’s 12 nodal displacements u_i^e using the shape function N_i :

$$u = \sum_{i=1}^{12} N_i(u_i^e). \quad (1)$$

The element mass matrix is then given by the integral over the element’s volume V^e :

$$m_{ij}^e = \int_{x \in V^e} \rho N_i(x) N_j(x) dV. \quad (2)$$

This is known as the consistent mass matrix, which accurately couples the motion of the element’s nodes. We employ the Gaussian numerical integration method, selecting t Gaussian integration points g_k within the tetrahedral element, with corresponding Gaussian integration weights w_k [8]:

$$m_e^{ij} = \rho V \sum_{k=1}^t w_k N_i(g_k) N_j(g_k). \quad (3)$$

Element Stiffness Matrix (k^e). The element stiffness matrix k^e represents the element’s resistance to deformation and is derived from the material’s elastic properties (Young’s modulus E and Poisson’s ratio ν), which defined as[8]:

$$k^e = \int_{x \in V^e} D(x)^T B(E, \nu) D(x) dV. \quad (4)$$

Here, $B(E, \nu)$ is the elasticity matrix representing the material model [1], and $D = \nabla_x N$ is a matrix derived from the shape functions with respect to the physical spatial coordinates.

With Lamé coefficients λ, μ defined as:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}, \quad (5)$$

we adopt the linear elastic model, where $B(E, \nu)$ is the matrix representation of the fourth-order isotropic elasticity tensor[11]. It relates the stress tensor σ to the displacement gradient, governed by Hooke’s Law:

$$\sigma_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \lambda \delta_{ij} \sum_k \frac{\partial u_k}{\partial x_k}. \quad (6)$$

$B(E, \nu)$ is computed as the Jacobian of the stress tensor σ with respect to the displacement gradient tensor F (where $F = \nabla u$). Mathematically, the entries of B are defined by the partial derivatives:

$$B_{ijkl} = \frac{\partial \sigma_{ij}}{\partial F_{kl}}. \quad (7)$$

In the implementation, this fourth-order tensor B is reshaped into a 9×9 matrix, mapping the vectorized displacement gradient to the vectorized stress.

We employ Gaussian integration method to calculate k^e , selecting t Gaussian integration points g_k within the tetrahedral element, with corresponding Gaussian integration weights w_k [8]:

$$k^e = V \sum_{k=1}^t w_k D(g_k)^T B(E, \nu) D(g_k). \quad (8)$$

2.1.2. Global Matrix Assembly

Finally, the global M and K matrices are constructed. These are large, sparse matrices, initially filled with zeros, with dimensions $N_{dof} \times N_{dof}$, where $N_{dof} = 3n$ is the total number of degrees of freedom in the entire mesh.

The assembly process iterates over every element e in the mesh. For each element, its local k_e and m_e matrices are computed. The entries of these local matrices are then added into the corresponding positions within the global M and K matrices, based on a global-to-local index mapping of the element's nodes. This "superposition" step results in the final global mass M and stiffness K matrices, which are symmetric, positive-definite (or semi-definite), and represent the complete dynamic properties of the discretized 3D object.

We employ PyTorch [10] to efficiently batch calculate both the element mass matrix and element stiffness matrix. Subsequently, these element matrices are assembled into global Coordinate Format (COO) sparse matrices for further processing, according to [8].

2.2. Modal Reduction and Sound Synthesis

To synthesize the impact sound, we seek to numerically solve the discretized equation of motion involving the full damping matrix D [2]:

$$M\ddot{u} + D\dot{u} + Ku = f(t). \quad (9)$$

Solving this coupled high-dimensional system directly is computationally prohibitive. Our fundamental strategy is to decouple the system into a set of independent one-dimensional oscillators. To achieve this, we perform the Generalized Eigenvalue Decomposition (GED) on the stiffness matrix K and mass matrix M :

$$KU = MU\Lambda. \quad (10)$$

Here, $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_N)$ is the diagonal matrix of eigenvalues, and U is the matrix of generalized eigenvectors (mode shapes). These eigenvectors are normalized to satisfy the following orthogonality conditions [6]:

$$U^T MU = I, \quad U^T KU = \Lambda. \quad (11)$$

We then transform the nodal displacement vector u into modal coordinates q using the linear mapping $u = Uq$. Substituting this into Eq. (9) and pre-multiplying by U^T , we obtain the transformed system:

$$U^T MU\ddot{q} + U^T DU\dot{q} + U^T KUq = U^T f(t). \quad (12)$$

To ensure the diagonalization of the damping term, we employ the Rayleigh damping model, where the damping matrix is approximated as a linear combination of mass and stiffness: $D = \alpha M + \beta K$. Exploiting the orthogonality properties, the damping term transforms as:

$$U^T (\alpha M + \beta K) U = \alpha I + \beta \Lambda. \quad (13)$$

Consequently, the coupled system simplifies to a diagonalized form:

$$\ddot{q} + (\alpha I + \beta \Lambda)\dot{q} + \Lambda q = F_{modal}(t), \quad (14)$$

where $F_{modal} = U^T f(t)$. Since all matrices on the left-hand side are diagonal, the system is fully decoupled. This implies that the vibration of each mode is independent of the others.

For efficient computation, we utilize the ARPACK solver to calculate only the k smallest eigenvalues and their corresponding eigenvectors. It is important to note that the first six eigenvalues are theoretically zero (or near-zero numerically). These correspond to the six rigid-body modes (three translations and three rotations) of the unconstrained object. Since rigid-body motion does not induce deformation or acoustic waves, these six modes are discarded. The remaining non-zero eigenvalues represent the elastic deformation modes, starting from the fundamental frequency.

Finally, for each retained mode i , the equation reduces to a standard damped harmonic oscillator:

$$\ddot{q}_i + (\alpha + \beta \lambda_i)\dot{q}_i + \lambda_i q_i = u_i^T f(t). \quad (15)$$

This 1D equation can be solved analytically to obtain the modal contribution $q_i(t)$, and the final displacement is reconstructed by summing these contributions:

$$u(t) \approx \sum_{i=7}^k u_i q_i(t). \quad (16)$$

For an impulse excitation $f(t)$ at $t = 0$, the solution to each mode is a damped sinusoid:

$$q_i(t) = A_i e^{-\sigma_i t} \sin(\omega_{d,i} t), \quad (17)$$

where A_i is the amplitude of mode i , $\omega_{d,i}$ is the damped angular frequency, and σ_i is the decay rate of mode i . $\omega_{d,i}$ and σ_i are represented by Rayleigh damping coefficient α, β and undamped natural angular frequencies ω_i :

$$d_i = \frac{1}{2}(\alpha + \beta \omega_i^2) \quad (18)$$

$$\omega_{d,i} = \omega_i \sqrt{1 - \left(\frac{d_i}{\omega_i}\right)^2}. \quad (19)$$

3. Tasks Details

3.1. Data-Driven Sound Synthesis

Frequencies. We map the modal frequencies from the Hertz scale to the Mel scale. These Mel-scaled values are then linearly normalized to the range $[-1, 1]$, bounded by the audibility frequency range (20 Hz to 20 kHz).

Geometry Features. We employ an OCNN-based model [15] as the shape encoder. This encoder processes the input 3D geometric mesh and extracts a 128-dimensional geometric feature vector.

Material Features. We normalize the input material parameters—Young’s modulus, Poisson’s ratio, and density—to the range $[-1, 1]$. Specifically, density and Young’s modulus undergo logarithmic normalization within the ranges $[500, 10000]$ and $[5 \times 10^8, 5 \times 10^{11}]$, respectively. In contrast, Poisson’s ratio is linearly normalized within the range $[0.1, 0.4]$. These normalized parameters are subsequently projected into a 32-dimensional material feature vector via a Multi-Layer Perceptron (MLP).

Eigenvalue Prediction. We concatenate the extracted geometry and material features and feed them into a Sinusoidal Representation Network (SIREN [14]), which is trained to output the object’s first 64 non-zero scaled natural frequencies. We minimize the mean squared error (MSE) between the predicted scaled frequencies and their corresponding ground-truth values.

Implementation Details.

- **NeuralSound [7]:** We retrained the Vibration Solver component in NeuralSound on our dataset in 64 eigenvalues for 10 epochs (~ 1 day of training on a single NVIDIA 4090 GPU), by which point the training loss had converged. It leads to performance exceeding those reported in its original paper.
- **Ours:** Our proposed model was trained for 100 epochs (~ 1 day of training on $6 \times$ NVIDIA RTX 4090 GPUs) using the batch size of 16. We use AdamW optimizer, with a learning rate of $1e-3$, and a weight decay of $1e-4$. The learning rate is decayed linearly to zero over the training period.

Performance details. The Time cost for NeuralSound, FEM (LOBPCG), and ours was benchmarked on a single NVIDIA 4090 GPU. Since FEM (ARPACK) is not a GPU-based method, it was tested on an Intel Xeon Platinum 8260 CPU.

3.2. Sound-Guided Shape Reconstruction

We adapt the Variational Autoencoder (VAE) architecture from Step1X-3D [9] for our framework. However, diverging from the original objective of self-reconstruction, we repurpose the architecture into a Conditional VAE (C-VAE). Specifically, our model is designed to reconstruct fine-grained structural details from coarse voxel inputs, conditioned on audio eigenvalues (modal frequencies) and material parameters. The pipeline consists of the following five stages:

- **Coarse Voxel Generation.** We discretize the normalized $[-1, 1]^3$ spatial domain into a grid of resolution $16 \times 16 \times 16$. For each mesh in our dataset, we evaluate the grid cells; if the center of a cell lies within the mesh boundary, the cell is filled with a cubic voxel, otherwise it remains empty. These occupied voxels are then tessellated into an OBJ triangular mesh, which serves as the initial coarse geometric condition.
- **Input Processing.** We uniformly sample a point cloud of $N = 32,768$ points from the surface of the generated coarse voxel mesh. This point cloud serves as the input to the VAE encoder, which maps the geometric data into a geometry latent representation with dimensions 1024×768 .
- **Condition Injection.** The 64-dimensional modal frequencies and the 3-dimensional material parameters are first scaled to the range $[-1, 1]$, following the protocol detailed in Sec. 3.1. Each modality is then processed by a dedicated Multi-Layer Perceptron (MLP) to project them into 768-dimensional embeddings: an audio feature vector and a material feature vector. These embeddings are concatenated with the geometry latent sequence, resulting in a composite feature tensor of size 1026×768 .
- **Mesh Decoding.** The composite feature tensor is fed into the Decoder Transformer block. Through the self-attention mechanism, the geometry tokens attend to the audio and material tokens, which allows the model to refine the geometry with physically consistent details based on the conditions. The decoder ultimately outputs an implicit Signed Distance Function (SDF) shape representation.
- **Supervision.** We pre-compute the ground truth SDF values for the meshes in our dataset on a dense 128^3 voxel grid. During training, we randomly sample 16,384 query points per mesh. We optimize the network by minimizing the Mean Squared Error (MSE) between the predicted SDF values (decoded from the coarse voxel input) and the ground truth SDF values at the sampled coordinates.

Implementation Details.

- **DiffSound [8]:** DiffSound performs instance-specific optimization on a Deep Marching Tetrahedra (DMTet) [12]

grid with a resolution of 64. Each mesh is optimized for 200 iterations, a duration empirically determined to be sufficient for the audio loss to converge. However, this approach incurs significant computational overhead, as it necessitates a Generalized Eigenvalue Decomposition (GED) at every optimization step. Due to these computational constraints, we restrict our comparative evaluation to a subset of random selected 100 test meshes.

- **Ours:** we trained the VAE from scratch on our training set. The model was configured with a global batch size of 48 and a decoder depth of 12 layers. All other hyper-parameters followed the default settings provided in the original Step1X-3D implementation. The model was trained for 200 epochs. The entire training process took approximately two days on $6 \times$ NVIDIA RTX 4090 GPUs.

Performance details. The time cost for DiffSound and ours was benchmarked on a single NVIDIA RTX 4090 GPU.

3.3. Cross-Modal Retrieval

To enable the sound, 3D shape, and image modalities to interact effectively, we design a cross-modal retrieval framework that learns a shared embedding space for all three modalities. This shared space allows for direct comparison and retrieval across different data types.

Modal Encoders. As discussed before, we employ specialized encoders to project each modality into a shared 128-dimensional feature space: a SIREN network [14] for sound, a VGG-16 [13] for images, and an Octree CNN [15] for 3D shapes. Additionally, for 3D shape encoding, we also take one-hot material category vector as input, which is then expanded to 128-dimension by a MLP and added to the embedding from OCNN.

Shared Embedding Space. To align the features from different modalities into a common space, we introduce modality-specific projection heads. We employed InfoNCE loss (Equation. 9) to train the encoders and projection heads jointly. This loss function encourages the model to bring related samples (e.g., an image and its corresponding sound) closer together in the embedding space while pushing unrelated samples apart.

Material Classification Header We incorporate an auxiliary material classification task to enhance the model’s understanding of material properties across modalities. A dedicated MLP-based classification head takes the embedding in the shared space as input and predicts the material category of the object. We use cross-entropy loss for this classification task, which is combined with the InfoNCE loss

to form the overall training objective. The material classification loss is weighted by a factor of 0.1 to balance its influence during training.

Training details. The model is trained for 200 epochs using a batch size of 24. We use AdamW optimizer, with a learning rate of $1e-3$, and a weight decay of $1e-4$, both of which are default settings in PyTorch. The learning rate is decayed linearly to zero over the training period. The model is trained on $6 \times$ NVIDIA A6000 GPUs, taking approximately 1 day to complete.

Comparison to Other works. We evaluate the model’s performance using standard retrieval metrics, including Top-1, Top-5, and Top-10 accuracy. These metrics assess the model’s ability to correctly retrieve the relevant item from a pool of candidates based on a given query from another modality.

While we use the same evaluation metrics as ReallImpact and ObjectFolder [3, 4], it is important to note that our task setup differs significantly, resulting in a direct numerical comparison infeasible. First, the data domains are distinct: while ReallImpact predominantly utilize real-world audiorecordings, and ObjectFolder series focuses on event-driven multisensory perception, our audio data is derived entirely from physical equation-based synthesis. Second, our method synthesizes a location-invariant “averaged” impact sound by applying a unit impulse excitation $\delta(t)$ to each vibrational mode, rather than derived from impacts at specific surface locations. Finally, the complexity of the retrieval task varies due to the scale of the search space. For instance, the Cross-Sensory Retrieval task in the ObjectFolder Benchmark [5] is evaluated on a test set of 100 meshes. In contrast, our evaluation requires retrieving the correct target from a dataset of over 4,000 candidates, which presents a much more challenging scenario.

4. Additional Task: Sound-Guided Solid Identification

Visual modalities alone lack the information density required to infer internal physical properties, such as differentiating between solid and hollow structures. Impact sounds, however, offer extrinsic evidence of these internal characteristics. As a supplementary experiment, we formulate a binary classification task that leverages both single-view images and the modal frequencies of impact sound to identify structural solidity.

Training Data. Adopting the hollow mesh generation methodology proposed in DiffSound [8], we generate hollow counterparts of existing solid objects by removing their interiors.

Specifically, according to [8], we define the “thickness” of generated hollow mesh based on the Signed Distance Function (SDF) of its corresponding solid counterpart. Let s_{\min} denote the global minimum SDF value, corresponding to the internal point strictly furthest from the surface (i.e., the maximum depth). We define a hollow object with a relative thickness ratio t as the set containing all points whose distance to the surface is within $t \cdot |s_{\min}|$. Consequently, a point P is considered to be inside the hollow shell if and only if its SDF value, $\mathcal{S}(P)$, satisfies the condition:

$$t \cdot s_{\min} < \mathcal{S}(P) < 0 \quad (20)$$

For our dataset creation, we synthesize hollow meshes by uniformly sampling the thickness ratio t from the range $[0.3, 0.7]$. Crucially, this process preserves the exterior mesh, rendering the hollow and solid objects visually indistinguishable.

We synthesized 1,000 hollow objects alongside their modal frequencies. By combining these with the existing solid counterparts from the original dataset, we constructed a balanced dataset of 2,000 samples. Each data entry comprises multi-modal inputs (audio and image) and a binary label (solid or hollow). Finally, the dataset was partitioned into a training set of 1,600 samples and a test set of 400 samples.

Methods. Following the design in Sec. 3.3, we employ a SIREN-based [14] sound encoder and a VGG-based [13] image encoder. Specifically, the input image and modal frequencies are processed by their respective encoders to extract visual and auditory embeddings. These feature vectors are subsequently concatenated and passed through a Multi-Layer Perceptron (MLP) to generate a two-dimensional output, representing the logits for the solid and hollow classes. The model is optimized using cross-entropy loss. We train the model for 100 epochs in training set, which takes approximately 3 hours on a NVIDIA RTX 4090 GPU.

Experiment Results. The classification accuracy on the test set is presented below. Furthermore, we evaluate the performance separately on the two distinct data sources: Objaverse and Generated items. The results demonstrate that our dataset enables data-driven approaches to effectively recognize internal object structures by leveraging audio cues.

	Objaverse	Generated	All
Accuracy \uparrow	74.00%	91.33%	87.00%

References

- [1] Jernej Barbic. Siggraph 2012 course notes fem simulation of 3d deformable solids: A practitioner’s guide to theory, discretization and model reduction. part 2: Model reduction (version: August 4, 2012). 2

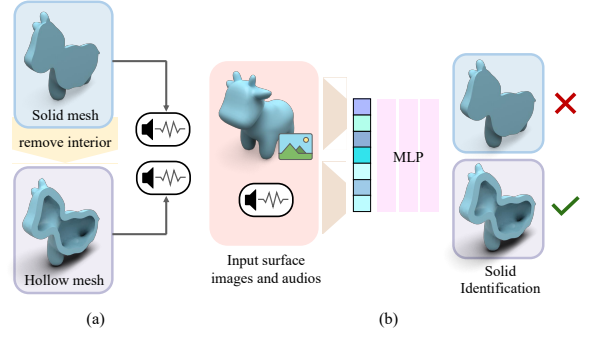


Figure 1. Sound-Guided Solid Identification. (a) For solid objects, we construct hollow counterparts by removing their interior. We then synthesize audios for both the solid and hollow objects. (b) Taking the surface rendering and modal frequencies as inputs, the model classifies whether the source object is solid or hollow.

- [2] Indrajit Chowdhury and Shambhu P Dasgupta. Computation of rayleigh damping coefficients for large systems. *The Electronic Journal of Geotechnical Engineering*, 8(0):1–11, 2003. 3
- [3] Samuel Clarke, Ruohan Gao, Mason Wang, Mark Rau, Julia Xu, Jui-Hsien Wang, Doug L James, and Jiajun Wu. Re-impact: A dataset of impact sound fields for real objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1516–1525, 2023. 5
- [4] Ruohan Gao, Yen-Yu Chang, Shivani Mall, Li Fei-Fei, and Jiajun Wu. Objectfolder: A dataset of objects with implicit visual, auditory, and tactile representations. In *CoRL*, 2021. 5
- [5] Ruohan Gao, Yiming Dou, Hao Li, Tanmay Agarwal, Jeanette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. The objectfolder benchmark: Multisensory learning with neural and real objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17276–17286, 2023. 5
- [6] Doug L James. Physically based sound for computer animation and virtual environments. In *ACM SIGGRAPH 2016 Courses*, pages 1–8, 2016. 3
- [7] Xutong Jin, Sheng Li, Guoping Wang, and Dinesh Manocha. Neursound: learning-based modal sound synthesis with acoustic transfer. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022. 4
- [8] Xutong Jin, Chenxi Xu, Ruohan Gao, Jiajun Wu, Guoping Wang, and Sheng Li. Diffsound: Differentiable modal sound rendering and inverse rendering for diverse inference tasks. In *SIGGRAPH ’24: ACM SIGGRAPH Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 2, 3, 4, 5, 6
- [9] Weiyu Li, Xuanyang Zhang, Zheng Sun, Di Qi, Hao Li, Wei Cheng, Weiwei Cai, Shihao Wu, Jiarui Liu, Zihao Wang, et al. Step1x-3d: Towards high-fidelity and controllable generation of textured 3d assets. *arXiv preprint arXiv:2505.07747*, 2025. 4

- [10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. [3](#)
- [11] Junuthula Narasimha Reddy. An introduction to the finite element method. *New York*, 27(14), 1993. [2](#)
- [12] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021. [4](#)
- [13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [5](#), [6](#)
- [14] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. [4](#), [5](#), [6](#)
- [15] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph. (SIG-GRAPH)*, 36(4), 2017. [4](#), [5](#)