PAVAS: Physics-Aware Video-to-Audio Synthesis

Supplementary Material

Contents

A. Supplementary Video

This work focuses on Video-to-Audio (V2A) generation, which is best viewed in video format. Please refer to the attached **supplementary video**. The video contains qualitative comparisons between two of the latest state-of-the-art V2A models [6, 18] and our approach on the VGGSound [4] test set, highlighting the perceptual audio quality and physics consistency achieved by PAVAS. Beyond the generation results on VGGSound test set, the video also includes several controlled generation scenarios using in-the-wild video clips:

- *Controlled semantics*: a hammer-crashing scene where only the material appearance is changed, demonstrating that PAVAS adjusts the generated impact sound according to the object's material cues.
- *Controlled velocity*: manipulating the object's velocity while keeping mass and material information constant, showing that faster motion produces sharper and higher-energy impact transients.
- *Controlled mass*: varying the object's mass while holding its material and motion constant, demonstrating that heavier objects yield proportionally stronger impact sounds.

These examples further validate that PAVAS responds coherently to visual and physical cues, producing both perceptually plausible and physically consistent audio.

B. Can We Estimate Plausible Physics Values?

Reliable physical conditioning requires that the estimated object mass and velocity correspond to plausible real-world magnitudes. In this section, therefore, we evaluate both components of the Physics Parameter Estimator (PPE) to ensure that the physical values used for conditioning are sufficiently reliable for physics-aware audio synthesis.

B.1. Mass Estimation Ability

For mass estimation, we follow the evaluation protocol of NeRF2Physics [25] and test our estimator on 500 objects from the Amazon Berkeley Objects (ABO) dataset [7]. We report four standard metrics widely used in mass estimation work [17, 25]—Absolute Difference Error (ADE), Absolute Log-Difference Error (ALDE), Absolute Percentage Error (APE), and Minimum Ratio Error (MnRE). Given ground-truth mass m and predicted mass $\hat{m}$, they are defined as:

$$\text{ADE} = |m - \hat{m}|, \tag{1}$$

$$\text{ALDE} = |\ln m - \ln \hat{m}|, \tag{2}$$

$$\text{APE} = \left| \frac{m - \hat{m}}{m} \right|, \tag{3}$$

$$\text{MnRE} = \min\left(\frac{m}{\hat{m}}, \frac{\hat{m}}{m} \right). \tag{4}$$

Table S1 shows that our method achieves favorable performance across four metrics while using only a single-view input, unlike NeRF2Physics which requires multi-view observations. Moreover, although prior work [25] reported limited generalization when applying earlier Vision–Language Models (VLMs) [13] to physical reasoning, our results show that a modern VLM [1], when guided with carefully designed prompts, can serve as an effective mass estimator. See Sec. E.1 for an example of a detailed prompt for the mass estimation.

B.2. Velocity Estimation Ability

To evaluate velocity estimation performance, we utilize STARSS23 [16], which provides metric-depth (i.e., displacement) annotations for sounding and moving objects. We adopt this dataset as a proxy benchmark since no established dataset exists for open-vocabulary object velocity estimation.

Method	ADE ↓	ALDE ↓	APE ↓	MnRE ↑
Image2mass [17]	12.496	1.792	0.976	0.341
2D CNN	15.431	1.609	14.459	0.362
LLaVA [13]	17.328	1.893	1.837	0.306
NeRF2Physics [25]	8.730	**0.771**	1.061	**0.552**
Ours	**6.954**	0.809	**0.823**	0.529

Table S1. **Mass estimation on ABO-500 test set.** We follow the same evaluation protocol of prior work [17, 25]. Note that NeRF2Physics [25] requires multi-view images to estimate mass.

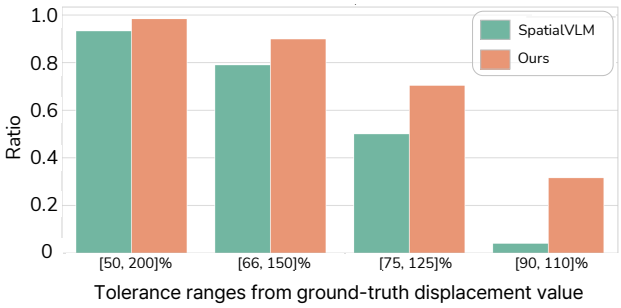

Figure S1. **Velocity estimation on STARSS23 test set.** We compare our metric-scale displacement estimation with the pseudo ground-truth dataset construction pipeline of SpatialVLM [3]. We measure the ratio of predictions falling within various tolerance ranges of the ground-truth displacement, providing a proxy evaluation of object-level velocity estimation accuracy.

We first convert each equirectangular video in STARSS23 into a normal field-of-view format using gnomonic projection [22]. Grounded-SAM2 [15] is then applied with the dataset's sound-class labels to obtain text-conditioned object masks. Low-quality segmentations are manually filtered out, resulting in 256 ten-second video clips. Using these filtered videos and masks, our dynamic 3D reconstruction model combined with inverse projection and centroid aggregation recovers metric-scale 3D trajectories for each object.

To assess the reliability of our metric-scale velocity values, we compare our pipeline not with the reasoning capability of SpatialVLM [3] but with its pseudo ground-truth dataset construction pipeline, which produces more accurate metric-scale displacement values using pretrained expert models. This setup evaluates the quality of metric-scale 3D lifting itself rather than the reasoning or VQA components of SpatialVLM, and it constitutes a more challenging evaluation than comparing against SpatialVLM's predicted outputs. We reproduce this pipeline using the open-source implementation officially acknowledged by the authors[1] and use the same filtered segmentation masks to ensure a fair comparison. We report the proportion of predictions that fall within several tolerance ranges of the ground-truth displacement (e.g., within $[50, 200]\%$). Figure S1 shows that our method achieves higher accuracy across all tolerance ranges,

[1]https://github.com/remyxai/VQASynth

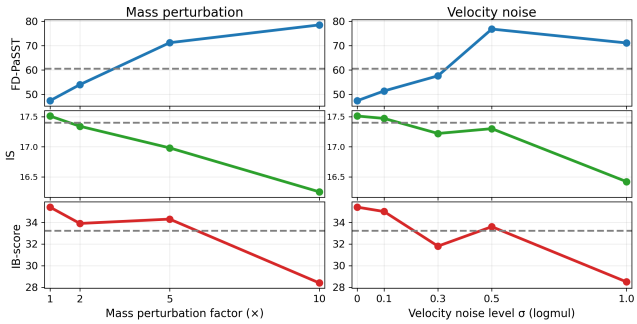

Figure S2. **Sensitivity of PAVAS-L to perturbed physics estimates at inference time. [Left]** mass perturbation by multiplicative factors $\{2, 5, 10\}$. **[Right]** velocity perturbation with log-multiplicative noise $\sigma \in \{0.1, 0.3, 0.5, 1.0\}$. The dashed line denotes MMAudio-L [6]. PAVAS degrades gradually as the perturbation magnitude increases, showing robustness to moderate estimation errors.

with particularly large improvements under stricter thresholds. These results indicate that our method provides more reliable metric-scale displacement (*i.e.*, velocity) estimation.

C. Robustness to Imperfect Physics Estimates

In this section, we analyze how PAVAS behaves when the estimated physical cues are imperfect or partially missing. Since the mass and velocity estimates used in our pipeline are generally reliable under the adopted proxy evaluations, we interpret the following perturbation experiments as stress tests rather than common operating conditions. We evaluate both inference-time sensitivity to perturbed physical values and fallback behavior when all physics tokens are replaced with learnable occlusion tokens.

C.1. Sensitivity to Mass Perturbation and Velocity Noise

To test sensitivity to estimation errors, we perturb the inferred mass and velocity values at inference time. For mass, we multiply the estimated value by factors of $\{2, 5, 10\}$. For velocity, we inject log-multiplicative noise with $\sigma \in \{0.1, 0.3, 0.5, 1.0\}$. Figure S2 summarizes the resulting trends, where the dashed line indicates the MMAudio-L [6]. As the perturbation magnitude increases, the performance of PAVAS degrades gradually rather than abruptly, indicating that the proposed conditioning mechanism is robust to moderate estimation errors. Noticeable degradation appears only under severe perturbations, such as mass $\times 10$ or velocity noise with $\sigma = 1.0$, which are substantially more extreme than the typical estimation errors observed in practice.

C.2. Graceful Fallback under Missing Physics Cues

We further analyze the extreme case where valid physical cues are entirely unavailable. In this setting, all mass and

Method	FD_{PaSST} ↓	IS↑	IB↑
Backbone	60.6	17.4	33.2
PAVAS-L	47.4	17.5	35.4
Fallback	64.5	17.2	32.8

Table S2. **Fallback via occlusion-token replacement.** When all physics tokens are replaced with occlusion tokens at inference time, PAVAS-L degrades gracefully toward the appearance-driven backbone [6] rather than failing abruptly.

velocity tokens are replaced with their corresponding occlusion tokens at inference time, yielding the fallback variant in Table S2. The fallback model performs close to the appearance-driven backbone [6], indicating that the proposed design does not fail catastrophically when physics cues are missing. Instead, the model gracefully falls back to the underlying multimodal generation pathway. This behavior is consistent with the occlusion-token training strategy used during physics-aware fine-tuning, which encourages robustness to missing or ambiguous motion cues.

D. Audio-Physics Correlation Benchmark

D.1. VGG-Impact

To assess physical realism, we curate VGG-Impact, a subset of the VGGSound test split consisting of 10 impact-related sound classes and 272 impact moments, where object mass and motion directly influence the resulting audio. The selected sound classes are:

- *basketball bounce*
- *bowling impact*
- *hammering nails*
- *bouncing on trampoline*
- *opening or closing car doors*
- *striking pool*
- *striking bowling*
- *forging swords*
- *chopping wood*
- *door slamming*

We then apply our Physics Parameter Estimator (PPE) to obtain frame-wise object masks and their corresponding physical quantities (mass and velocity). Videos are manually filtered out under two conditions: (1) when they do not contain clear object–object interaction sounds (we exclude ambiguous scenes with unclear contact dynamics), and (2) when object masks fail to provide reliable tracks due to occlusion or segmentation errors.

After filtering, we retain video–audio pairs containing identifiable impact events, along with per-object mass and velocity sequences. These serve as the basis for evaluating whether generated audio reflects the underlying physical parameters via the proposed Audio–Physics Correlation Coefficient (APCC).

D.2. Audio-Physics Correlation Coefficient

To assess the physical plausibility of generated audio, we introduce the *Audio–Physics Correlation Coefficient (APCC)*, which measures how consistently variations in physical magnitude are reflected in the acoustic response. We extract impact onsets $\{\tau_j\}$ from the generated waveform using the SuperFlux onset detector [2]. The onset detection function (ODF) is computed on the mel-spectrogram $S_{\mathrm{mel}}(\tau, f)$ as a local spectral energy difference:

$$\mathrm{ODF}(\tau) = \sum_{f} \left[\, S_{\mathrm{mel}}(\tau, f) - S_{\mathrm{mel}}(\tau - \delta, f) \,\right]_{+}, \quad (5)$$

where $[\cdot]_+$ denotes half-wave rectification. The spectral onset energy used for correlation is therefore $y_j = \mathrm{ODF}(\tau_j)$.

To quantify the physical magnitude associated with each event, we measure the kinetic energy drop for the corresponding object o_i around each onset τ_j. Given its estimated mass m_i and velocity sequence $\{v_i^\ell\}_{\ell=1}^{L}$, we define the incoming and outgoing velocities as

$$v_{i,\mathrm{in},j} = \max_{\ell \in \mathcal{N}^-(\tau_j)} v_i^\ell, \qquad v_{i,\mathrm{out},j} = v_i^{\ell_j}, \quad (6)$$

where ℓ_j is the frame index closest to onset time τ_j, and $\mathcal{N}^-(\tau_j)$ denotes a short pre-onset neighborhood. The corresponding kinetic energy change is

$$\Delta E_{i,j} = \tfrac{1}{2}\, m_i \big((v_{i,\mathrm{in},j})^2 - (v_{i,\mathrm{out},j})^2 \big), \quad (7)$$

representing the mechanical energy lost at impact, which is expected to be radiated as an acoustic impulse [10].

For each video, we z-normalize $\{\Delta E_{i,j}\}$ and $\{y_j\}$ and compute a Pearson correlation per interaction class c, yielding APCC_c. We evaluate this correlation for both a ground-truth audio and a model-generated audio, obtaining $\mathrm{APCC}_{\mathrm{GT},c}$ and $\mathrm{APCC}_{\mathrm{model},c}$, respectively. The final score is the class-averaged deviation

$$\text{APCC-}\Delta = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \left| \mathrm{APCC}_{\mathrm{GT},c} - \mathrm{APCC}_{\mathrm{model},c} \right|. \quad (8)$$

A lower APCC-Δ indicates that the generated audio more closely matches the real coupling between kinetic energy changes and spectral onset strength.

D.3. Robustness of APCC

We further analyze whether APCC-Δ reflects meaningful physics–audio coupling rather than spurious correlation. Under inter-class shuffling of the estimated physical magnitude changes, APCC-Δ becomes substantially worse (0.378 → 0.655), indicating that the metric does not arise from chance correlation. We additionally inject noise into the estimated physical quantities and observe a progressive degradation in APCC-Δ as the perturbation magnitude increases (see Fig. S3). Together, these results support that APCC-Δ captures meaningful physics–audio coupling and responds con-

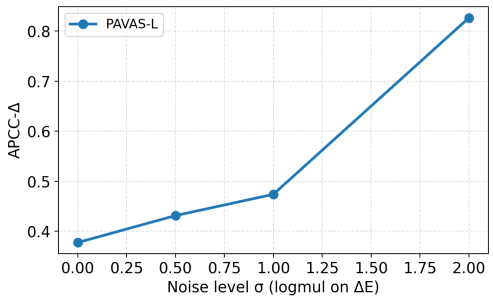

Figure S3. **APCC-Δ under injected noise.** As the perturbation magnitude increases, APCC-Δ degrades progressively, supporting that the metric is sensitive to meaningful changes in the estimated physical quantities.

sistently to perturbations in the estimated physical quantities.

E. Module Details

E.1. Instruction Prompts for VLMs

We employ a Vision–Language Model (VLM) [1] to extract physics-relevant object information through two sequential stages: (1) *moving-object discovery*, which identifies objects that exhibit genuine physical motion throughout the video, and (2) *mass estimation*, which predicts a physically plausible mass (in kilograms) for each moving object. Both stages rely on instruction prompts designed to produce structured, machine-readable JSON output, suitable for downstream processing in the Physics Parameter Estimator (PPE). Below, we detail the motivation, design principles, and full templates used for each stage.

Prompt for moving-object discovery. The first stage requires identifying which entities in the scene undergo meaningful motion over the entire clip. Because apparent frame-to-frame changes may arise from camera movement or aliasing, the prompt explicitly instructs the VLM to reason at the sequence level and to consider only true object motion driven by translation, rotation, articulation, or deformation. To guarantee reliable detection, the prompt includes: (i) a taxonomy of admissible motion types, (ii) explicit exclusion rules for non-solid continuous media (*e.g.*, water, smoke, fire), and (iii) strict formatting constraints (plain JSON, no markdown, no invented object names). The VLM is required to return a JSON object mapping each detected moving object to a short verb phrase describing its action (e.g., `"basketball":"bouncing"`, `"door":"opening"`). A full prompt template is provided in Table S7.

This structured output plays a dual role: it filters out irrelevant scene elements and provides canonicalized object keys that are passed verbatim to the subsequent mass-estimation stage, ensuring consistency across stages and preventing object-identity drift.

Prompt for mass estimation. Given the JSON map of detected moving objects generated by the previous prompt, the second instruction estimates the mass of each object. The prompt explicitly injects this JSON as the `MOVING OBJECTS` block and obligates the VLM to: (i) use the provided keys without alteration, (ii) return a single plain JSON object, and (iii) supply, for every object, both a concise rationale and a numeric field `"weight_kg"`.

The rationale must be grounded in visual evidence—such as material composition, approximate dimensions, density cues, or human body build—and must end with the same numeric estimate that appears in `"weight_kg"`. The policy section enforces strict structural constraints (no renaming, no additional keys, no markdown, numeric values only) and offers heuristics for estimating mass from typical object specifications. Several few-shot examples are embedded directly in the prompt to increase output stability and reduce hallucination. A template is given in Table S8.

Together, these two prompts enable the VLM to produce consistent, interpretable, and physically grounded object descriptors that serve as input to our Physics Parameter Estimator. The strict formatting guarantees ensure that the downstream modules can parse and align object identities reliably, allowing robust integration of semantic and physical cues into the overall PAVAS pipeline.

E.2. Details of Physics Parameter Estimator

The Physics Parameter Estimator (PPE) extracts object-level physical attributes from unconstrained visual scenes using three components: (i) a Vision–Language Model (VLM) for sequence-level motion discovery and object-wise mass estimation, (ii) an open-vocabulary video segmentation pipeline based on Florence-2 [23] and SAM2 [15], and (iii) a CUT3R [21]-based dynamic 3D reconstruction module for metrically scaled velocity estimation. All modules operate on the same per-video object keyspace produced by the VLM, ensuring that object descriptions, segmentation tracks, and 3D trajectories remain consistently aligned throughout the pipeline.

Vision–Language Model. Both moving-object discovery and mass estimation are implemented with the `Qwen/Qwen2.5-VL-7B-Instruct` model[2]. Input videos are provided through the chat interface as a single video message with adaptive spatial downsampling controlled by a vision-token budget of `MAX_PIXELS`=512×28^2. In both stages, generation is bounded by `max_new_tokens`=192 and `min_new_tokens`=16.

For moving-object discovery, decoding is performed without stochastic sampling (default greedy decoding) to stabilize key naming and JSON structure. For mass estimation, we reuse the same model and pixel scaling but enable stochastic

[2]https://huggingface.co/Qwen/Qwen2.5-VL-7B-Instruct

decoding with `do_sample=True`, `temperature`=0.4, and `top_p`=0.92 to allow more nuanced and diverse physical reasoning output. A truncation heuristic detects incomplete JSON (e.g., unbalanced braces or outputs that terminate near the token limit) and triggers a single regeneration with an expanded budget (`max_new_tokens` multiplied by `RETRY_MULTIPLIER=2`). In both stages, we parse the model output as a single JSON object. For moving-object discovery, we further clean the map by lowercasing and discarding clearly invalid entries, whereas for mass estimation we ensure there is exactly one record per input object and convert the `"weight_kg` fields into numeric values.

Open-vocabulary video instant segmentation model. To obtain open-vocabulary instance masks for each moving object, we couple Florence-2 [23] with the SAM2 [15] video segmentation model. For every moving-object key discovered by the VLM [1], we construct a text grounding query in the format used by Florence-2 for open-vocabulary detection (*i.e.*, by prefixing the object name). Florence-2 applies this query to sparsely sampled keyframes, where the sampling interval is chosen so that the resulting keyframes align with the 8 Frames Per Second (FPS) temporal grid used by the downstream 3D reconstruction pipeline (*i.e.*, CUT3R).

Each detected box is converted into a binary instance mask using the SAM2 image predictor, and these masks initialize the SAM2 video predictor, which leverages XMem-style propagation [5] to track object instances, producing temporally consistent mask sequences. When Florence-2 returns no detection for a keyframe, we insert an empty mask to preserve alignment across time. The final output is a per-object sequence of instance masks, providing a stable, frame-aligned segmentation track that directly supports subsequent 3D point-cloud fusion and instantaneous velocity estimation.

Dynamic 3D reconstruction model. We obtain metrically scaled 3D geometry using CUT3R [21], instantiated with the `cut3r_512_dpt_4_64.pth` checkpoint[3]. All videos are temporally resampled to 8 FPS, which aligns the inference of CUT3R with the segmentation sequence. CUT3R operates on a resolution of 512×512; accordingly, RGB frames are resized to this resolution prior to inference. The instance masks produced by SAM2 are also resized to the same resolution of CUT3R, which preserves discrete object boundaries while avoiding interpolation artifacts that could distort the mask geometry.

For each frame, CUT3R predicts a dense metric point map together with a per-pixel confidence score. Points whose confidence is below a fixed threshold (0.7) are removed, suppressing geometrically unstable or low-parallax regions while retaining reliable surfaces. The remaining points form a metrically meaningful reconstruction aligned to CUT3R's estimated camera coordinate system. Then, object-wise 3D geometry is obtained by indexing the reconstructed point map with each resized instance mask. This extracts all 3D points whose originating pixels were assigned to a given object. If no valid 3D points remain after confidence filtering, the object is treated as absent for that frame; otherwise, a centroid is computed by averaging the surviving points.

Centroid trajectories are defined over the same 8 FPS temporal grid. Instantaneous velocities are computed via finite differences of successive centroids, *i.e.*, Euclidean displacement divided by the frame interval ($\Delta t = 1/8$ second). No temporal smoothing or post-hoc filtering is applied, so the reported velocities directly reflect CUT3R's raw geometric estimates. Frames with missing centroids propagate a `None` placeholder. During downstream conditioning, any velocity that depends on such missing endpoints is replaced with a learnable velocity-occlusion token, ensuring that occluded motion segments are explicitly encoded and do not contaminate the continuous velocity embedding.

E.3. Handling Missing Temporal Observations

PAVAS uses two learnable occlusion tokens for temporally missing observations. First, in object feature extraction, frames where the object is absent or occluded are replaced with an *object-occlusion token*, yielding a stable object-level conditioning stream across time. Second, in velocity modulation, when a valid velocity cannot be computed for a frame because the required masks or 3D endpoints are unavailable, the corresponding input is replaced with a *velocity-occlusion token*. Both occlusion tokens are learnable embeddings with the same dimensionality as their corresponding conditioning streams, allowing missing observations to be handled without changing the conditioning interface. No mass-occlusion token is introduced, since mass is estimated as a time-invariant scalar for each object rather than a frame-wise quantity. These tokens therefore serve different roles: the object-occlusion token handles missing object observations, whereas the velocity-occlusion token handles unavailable motion estimates.

E.4. Diffusion Transformer Backbone

Our model adopts a diffusion–transformer architecture that integrates audio, vision, text, and synchronization cues within a unified latent space. The backbone consists of (i) a sequence of diffusion transformer blocks that jointly process tokenized audio, visual, and textual inputs, and (ii) a conditioning pathway that delivers semantic, visual, and physics-aware signals to every block. Audio is represented as a latent sequence produced by a pretrained Variational Auto Encoder (VAE) [9], enabling efficient diffusion while preserving high-fidelity reconstruction. This section describes the backbone configuration, the formation of the multimodal conditioning stream $\mathbf{c}_{\text{multi}}$, and the physics-specific pathways $\mathbf{c}_{\text{mass}}$ and $\mathbf{c}_{\text{vel}}$. See Fig. S4 for a detailed backbone structure.

[3] https://github.com/CUT3R/CUT3R

Model	Sample rate	Latent dim	Hidden dim h	(N_M, N_U)	Params (M)
PAVAS-S	16 kHz	20	448	(4, 8)	166
PAVAS-M	44.1 kHz	40	896	(4, 8)	630
PAVAS-L	44.1 kHz	40	896	(7, 14)	1039

Table S3. **Summary of model variants.** Each variant differs in latent dimensionality, hidden width, and the number of multimodal (N_M) and audio-only (N_U) transformer blocks.

Audio latent representation. Audio modality is modeled in a compact latent domain obtained through a two-stage spectral encoder. Waveforms are first converted into mel spectrograms via short-time Fourier analysis. For the 16 kHz model, we use 80 mel bins with a hop size of 256 samples, yielding a latent frame rate of 31.25 FPS. For the 44.1 kHz model, we use 128 mel bins with a hop size of 512 samples, resulting in a frame rate of 43.07 FPS. Each mel frame is then encoded by a pretrained VAE into a 20-dimensional (16 kHz) or 40-dimensional (44.1 kHz) latent vector using a 1D convolutional encoder–decoder with a temporal downsampling factor of 2. The VAE adopts magnitude-preserving convolutional and normalization layers, which stabilize latent statistics without affecting perceptual quality.

The diffusion backbone operates directly on these latent sequences, benefiting from both reduced temporal length and a smoother optimization landscape. During inference, generated latents are decoded back into mel spectrograms via the same VAE and converted to waveforms by a neural vocoder [11]. These latent representation provide high fidelity while remaining computationally efficient for diffusion-based generation.

Model variants. We train three model capacities that vary in hidden width and depth while sharing the same multimodal block design. The 16 kHz model (PAVAS-S) operates on 20-dimensional audio latents with hidden size h=448 and uses a stack of $(N_M, N_U) = (4, 8)$ multimodal and audio-only transformer blocks. The higher-rate variants (PAVAS-M/L) operate on 40-dimensional latents at 44.1 kHz and increase the hidden width to h=896 to accommodate the doubled latent dimension. The largest model (PAVAS-L) further increases depth to $(N_M, N_U) = (7, 14)$, offering the highest generation fidelity. A summary of architectural differences is provided in Table S3.

Table S4 shows that increasing model capacity consistently improves physics correlation and perceptual quality. PAVAS-M achieves the highest audio quality, while PAVAS-L yields the strongest physics alignment (APCC-Δ) and the rest of perceptual metrics, establishing it as our default model for the main experiment.

Input projection layers. Before joint fusion, each modality is projected into the shared hidden dimension:

- *Text:* token embeddings are linearly projected and refined by a lightweight MLP.
- *Vision:* CLIP patch features are projected and processed by a ConvMLP for local spatial mixing.

Model	Physics corr.	Distribution match.	Audio quality	Semantic align.	Temporal align.
	APCC-$\Delta\downarrow$	$\text{FD}_{\text{PaSST}}\downarrow$	IS$\uparrow$	IB-score$\uparrow$	DeSync$\downarrow$
PAVAS-S	0.412	65.67	16.50	29.41	0.448
PAVAS-M	0.395	49.08	**17.94**	34.54	0.449
PAVAS-L	**0.378**	**47.38**	17.51	**35.41**	**0.446**

Table S4. **Quantitative results of PAVAS model variants on VGGSound test split.** We report physics correlation (APCC-Δ), FD_{PaSST}, and the perceptual metrics (audio quality, semantic, and temporal alignment). PAVAS-L shows the strongest physics alignment and overall most favorable performance across metrics.

- *Synchronization cues:* features from the synchronization encoder [8] pass through a 1D convolution, activation, and ConvMLP.
- *Audio latents:* VAE-encoded spectrogram latents undergo a temporal convolution and ConvMLP.

Temporal layout of synchronization features. The synchronization encoder [8] outputs short feature sequences extracted from overlapping video clips. Since clip-local positions are ambiguous, we introduce a learnable positional embedding per clip window. After adding this embedding, clip windows are flattened into a single sequence and temporally interpolated to match the audio latent resolution.

Construction of multimodal conditioning $\mathbf{c}_{\text{multi}}$. At each diffusion timestep t, the backbone receives a multimodal context vector that summarizes all non-physical conditioning signals. We begin by forming global text and vision descriptors by average-pooling the projected CLIP text and visual token sequences, and feeding their sum into a MLP:

$$h_{\text{glob}} = f_{\text{MLP}}\big(\text{pool}(h_{\text{text}}) + \text{pool}(h_{\text{vision}})\big).$$

This global feature is then combined with the diffusion timestep embedding $f_{\text{time}}(t)$ to produce a sequence-invariant context $g(t)$, which is broadcast along the temporal dimension. The frame-aligned synchronization stream is upsampled to the audio frame rate, yielding $h_{\text{sync}}(t)$, and the final multimodal conditioning vector is formed as

$$\mathbf{c}_{\text{multi}}(t) = g(t) + h_{\text{sync}}(t).$$

Importantly, the upsampled synchronization features $h_{\text{sync}}(t)$ are injected *only* when conditioning the audio pathway; multimodal blocks use only the $g(t)$ for text and vision branches.

Multimodal and unimodal blocks. The model processes all token streams through N_M multimodal blocks, followed by N_U unimodal (audio path only) blocks. Each multimodal block performs:

1. adaptive normalization of each stream using parameters predicted from $\mathbf{c}_{\text{multi}}$, $\mathbf{c}_{\text{mass}}$, and $\mathbf{c}_{\text{vel}}$,
2. concatenated self-attention over the audio, vision, and text token sequences,
3. modality-wise splitting of the joint output back into audio, vision, and text streams,
4. per-stream feed-forward refinement.

Synchronization features influence the block only through

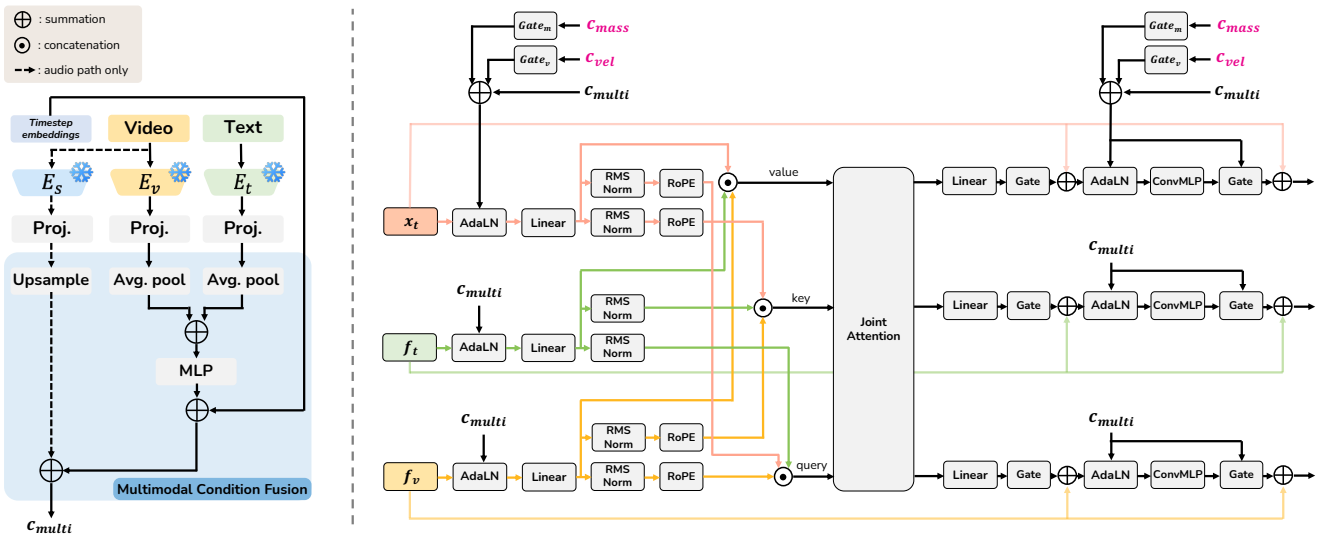

Figure S4. **Multimodal Diffusion Transformer Blocks and Conditioning Path. [Left]** The multimodal condition $\mathbf{c}_{\text{multi}}$ is constructed from synchronization, visual, and textual features together with the diffusion timestep embedding. Note that the upsampled synchronization features are used only when conditioning the audio pathway. **[Right]** Each multimodal diffusion transformer block contains three parallel streams (audio, text, video) that interact through *joint attention*. In contrast, the audio-only (unimodal) blocks used later in the network reuse the same audio pathway but replace joint attention with *self attention*; these blocks are not shown in the diagram but follow directly from the audio branch depicted here. $\mathbf{x}_t$, $\mathbf{f}_t$, and $\mathbf{f}_v$ denote the audio latent, text features, and video features, respectively. Physics conditions $\mathbf{c}_{\text{mass}}$ and $\mathbf{c}_{\text{vel}}$ are injected via Δ-modulation, where learnable gates ($\mathbf{Gate}_{\text{m}}$, $\mathbf{Gate}_{\text{v}}$) scale the mass and velocity signals and add them residually to the base multimodal condition $\mathbf{c}_{\text{multi}}$, enabling gradual and stable incorporation of physical effects.

$\mathbf{c}_{\text{multi}}$ and do not form a separate token sequence. Audio-only blocks retain the same conditioning but operate strictly on the audio latent sequence.

Δ-modulation vs. Direct summation. In the ablation study of Table 3 in the main paper, we consider two strategies of injecting physics conditions into Adaptive Layer Normalization (AdaLN) layer:

- *Δ-modulation (ours):* The base AdaLN parameters are predicted from the multimodal condition $\mathbf{c}_{\text{multi}}$. Physics conditions ($\mathbf{c}_{\text{mass}}$ and $\mathbf{c}_{\text{vel}}$) are fed to zero-initialized residual branches that predict only a *correction* to this base modulation. At initialization, these branches produce no effect, and their contribution grows gradually as training learns mass- and motion-dependent adjustments. Formally, each block refines its AdaLN parameters as $\tilde{\omega} = \omega(\mathbf{c}_{\text{multi}}) + \alpha_m g_m(\mathbf{c}_{\text{mass}}) + \alpha_v g_v(\mathbf{c}_{\text{vel}})$, where g_m, g_v are lightweight MLPs and α_m, α_v are learnable gates. We show this strategy in the Fig. S4.
- *Direct summation (ablative baseline):* In this variant, the physics streams are *added directly* to the multimodal condition, without a residual zero-initialized pathway. We form a physics-augmented condition $\mathbf{c}_{\text{phys}} = \mathbf{c}_{\text{multi}} + \mathbf{c}_{\text{mass}} + \mathbf{c}_{\text{vel}}$ and compute modulation parameters as $\tilde{\omega} = \omega(c_{\text{phys}})$. This corresponds to a straightforward feature-level mixing of mass and velocity with $\mathbf{c}_{\text{multi}}$, lacking the gradual, gated adaptation provided by Δ-modulation.

By comparing these two modes, we can disentangle the benefit of residual, gate-controlled physics injection from simply augmenting the multimodal condition with additional features.

E.5. Training Details

Overall procedure. PAVAS is trained in a two stage manner. The first stage trains a general-purpose multimodal latent diffusion backbone for video-to-audio generation. In the second stage, we introduce the Physics Parameter Estimator (PPE) and the Physics-Driven Audio Adapter (Phy-Adapter) and fine-tune the backbone to incorporate mass- and velocity-aware conditioning.

Stage 1: Backbone training. Unless stated otherwise, all backbone variants (S/M/L) share the same optimization hyperparameters. We use the AdamW optimizer with a learning rate of 1×10^{-4}, weight decay 1×10^{-6}, and $(\beta_1, \beta_2) = (0.9, 0.95)$. A linear warm-up of 1K steps is followed by a cosine decay schedule over 300K iterations, with learning rate reductions at 80% and 90% of training, reaching 1×10^{-5} and 1×10^{-6}, respectively. Mixed-precision (bf16) training and gradient clipping are applied to improve numerical stability. We also maintain an exponential moving average (EMA) of the model weights using a post-hoc formulation with a relative width of $\sigma_{\text{rel}} = 0.05$. Audio latents and visual/text embeddings are precomputed offline and streamed from disk during training.

Stage 2: Physics-aware fine-tuning. During physics integration, the audio, visual, and text encoders remain frozen, while the diffusion transformer blocks, the gating branches,

and the physics-conditioning pathways are updated. We reuse the same optimizer configuration but shorten training to 30K iterations and reduce the learning rate to 1×10^{-5}. To ensure robustness when motion cues are ambiguous or missing, physics tokens (mass or velocity) are randomly substituted with their corresponding null tokens with probability 0.1. This dropout provides the model with implicit classifier-free denoising behavior for the physics pathway.

Batching and compute. All models are trained with a global batch size of 512 using distributed data-parallel training on NVIDIA H100 GPUs. To keep the compute setting consistent across stages, both stage 1 and stage 2 use the same hardware allocation: PAVAS-S is trained on 2 H100 GPUs, while PAVAS-M and PAVAS-L are trained on 8 H100 GPUs. We adopt `bf16` mixed-precision training to reduce memory footprint and to stabilize optimization at large batch sizes. For training efficiency, all audio latents and visual/synchronization features are precomputed offline and streamed from disk during training, avoiding on-the-fly STFT, VAE encoding, or vision forward passes. This caching strategy ensures that both 16 kHz and 44.1 kHz variants achieve similar throughput despite differences in latent dimensionality and backbone width, and it allows the larger PAVAS-L model to fit comfortably within the memory budget while maintaining stable training dynamics.

F. Runtime and Invalid/Missing-Observation Statistics of PPE

We report module-wise runtime profiling and representative statistics of the PPE stack. Runtime is measured on a single H100 GPU after warm-up and averaged over 100 randomly sampled videos from VGGSound. Although PAVAS introduces non-trivial computational overhead, it is not intended for real-time operation. These results instead characterize the cost of the current pipeline and how often invalid or missing observations arise in unconstrained videos.

F.1. Module-wise Runtime Profiling

Table S5 reports the average runtime of each component in PAVAS-L. Among the PPE modules, CUT3R [21] dominates the computational cost, whereas the Phy-Adapter itself adds only a small overhead relative to the backbone. This indicates that the main bottleneck lies in extracting physically grounded conditioning signals rather than injecting them into the generator.

F.2. Invalid and Missing-Observation Statistics

We also quantify representative cases of invalid estimates and missing observations in the PPE pipeline. Text–scene mismatches due to VLM hallucination occur in only 0.04% of samples, and invalid mass outputs occur in 3.55%. For velocity estimation, CUT3R [21] produces low-confidence 3D reconstructions in 1.12% of frames, which are replaced with occlusion tokens. In Grounded-SAM2 [15], 65.15% of frames are replaced with occlusion tokens. Importantly, this high ratio mainly reflects natural occlusions and fast motion in in-the-wild videos rather than segmentation failure. Overall, these statistics show that truly invalid estimates are rare, while many missing observations arise from realistic visual ambiguity that PAVAS explicitly handles through fallback tokens (see Sec. C.2).

VLM (mov.)	VLM (mass)	GSAM2	CUT3R	Phy-Adapter	Backbone
0.90	0.73	0.71	4.82	0.32	1.90

Table S5. **Module-wise runtime of PAVAS-L (seconds per 10-second video).** Runtime is measured on a single H100 GPU after warm-up and averaged over 100 randomly sampled videos from VGGSound. CUT3R [21] dominates the PPE runtime, while the Phy-Adapter adds relatively small overhead.

Metric	VGG-NonImpact		VGG-Impact	
	MMAudio-L	PAVAS-L	MMAudio-L	PAVAS-L
$FD_{PaSST}\downarrow$	62.2	**47.9**	203.4	**182.9**
$FD_{PANNs}\downarrow$	4.73	**3.92**	19.8	**18.4**
$FD_{VGG}\downarrow$	**1.01**	1.09	**4.10**	5.41
$KL_{PANNs}\downarrow$	1.66	**1.56**	1.67	**1.50**
$KL_{PaSST}\downarrow$	1.40	**1.36**	1.29	**1.19**
IS↑	17.5	**17.6**	**4.72**	4.45
IB-score↑	33.1	**35.4**	33.9	**36.0**
DeSync↓	**0.44**	0.45	**0.30**	0.31

Table S6. **Quantitative evaluation on non-impact and impact sets.** PAVAS-L improves over MMAudio-L on most metrics in both VGG-NonImpact and VGG-Impact, suggesting that the benefit of physics-aware conditioning is not limited to impact-centric scenes.

G. Generalization Beyond Impact Scenes

We additionally compare PAVAS-L with MMAudio-L [6] on both VGG-NonImpact and VGG-Impact to assess its performance beyond impact-centric scenes. VGG-NonImpact is constructed by excluding the 10 impact-related classes used in VGG-Impact from the VGGSound test split. Table S6 shows that PAVAS-L improves over MMAudio-L on most metrics in both settings, suggesting that the benefit of physics-aware conditioning extends beyond impact-heavy events while remaining effective on impact-centric scenes.

H. Setup of User Study

To complement the objective evaluations, we conduct a user study to assess the perceptual quality of the generated audio. We compare PAVAS-L against six state-of-the-art video-to-audio models—See & Hear [24], V-AURA [19], VATT [14], V2A-Mapper [20], TARO [18], and MMAudio-L [6]—selected for their strong performance across semantic, temporal, and distributional metrics. We sample eight non-speech clips from the VGGSound test set, excluding low-resolution or ambiguous videos. For each clip, participants evaluate seven model outputs played back-to-back

over the same visual content, with the ordering randomized to avoid bias. A total of 27 participants take part in the study, resulting in 1,512 individual ratings.

Participants rate each model output independently using a 5-point Likert [12] scale (1–5; strongly disagree to strongly agree). After each audio segment finishes, they pause the video and answer four questions corresponding to the following evaluation criteria:

- **Audio Quality.** *"Rate whether the audio sounds clear, natural, and free of distracting noise or artificial artifacts. Ignore the visual content and judge the audio alone."*
- **Semantic Alignment.** *"Rate whether the type of sound matches the events or actions shown in the video (*e.g.*, whether the produced sound is appropriate for the depicted scenario)."*
- **Temporal Alignment.** *"Rate whether audio events occur at the correct time relative to visible events (*e.g.*, impacts, collisions, or actions). Misalignment includes delayed, early, or repeated events."*
- **Physical Plausibility.** *"Rate whether the audio reflects the physical properties observed in the video—such as mass, material, speed, and impact strength. Sounds should feel physically consistent with the visual motion and contact forces."*

Table 2 in the main paper summarizes the results. PAVAS-L achieves the highest mean score across all four aspects, with particularly strong gains in the newly introduced physical plausibility dimension.

I. Additional Discussion on Occlusion and Off-Screen Audio

PAVAS does not assume that sounding objects remain perfectly visible and trackable throughout the video. In practice, missing object or motion cues often arise due to natural occlusions, fast motion, or unreliable reconstruction in unconstrained videos. To handle such cases, the model uses occlusion tokens as fallback inputs rather than requiring complete physical observations at every frame. Moreover, PAVAS can still generate off-screen sound sources through its text-conditioned generation pathway, similar to prior V2A systems that use text prompts (*e.g.*, MMAudio [6]). This design allows the model to benefit from explicit physical conditioning when reliable cues are available, while remaining robust when such cues are partially missing.

J. Additional Qualitative Samples

To complement the quantitative results in the main paper, we provide additional qualitative comparisons in Figs. S5 and S6, highlighting how PAVAS differs from recent Video-to-Audio (V2A) generation models [6, 14, 18, 19, 24]. For each example, we visualize mel-spectrograms generated by state-of-the-art baselines alongside our method and the ground-truth audio. We annotate each figure with (i) green dashed lines to indicate spectral structures that correspond to visually observable events, and (ii) icon markers denoting audible objects or interactions present in the scene.

Across the qualitative samples, ours better matches the timing, duration, and spectral shape of audio events depicted in the video. In the goose honking and golf-driving examples (Fig. S5), competing methods often produce temporally duplicated impacts or unnaturally prolonged vocalizations. In contrast, ours aligns its transient events with the visual dynamics and avoids over-extending acoustic energy when the visual cue indicates only a short-lived action. Figure S6 further demonstrates this. In the dog-barking scene, existing V2A models fail to emit a bark at the moment when the dog opens its mouth, or they produce temporally shifted patterns. PAVAS instead generates a bark whose onset and spectral structure closely follow the ground truth. A similar observation holds in the wood-chopping example: our method produces sharply localized, short-decay broadband transients characteristic of real impacts, whereas other methods generate misaligned strikes or smeared spectral patterns.

Overall, these qualitative examples illustrate how incorporating physics-aware cues into the diffusion backbone helps PAVAS generate audio that is not only temporally and semantically consistent with the video, but also physically plausible in terms of impact strength, material response, and motion-dependent timing.

Table S7. **Prompt used for moving-object discovery in the Physics Parameter Estimator (PPE).**

Return only a plain JSON object (no markdown, no code fences, no extra text) that maps the moving objects in the video to their visible state or action.

Definition of a moving object (sequence-level):
- Determine motion over the entire clip (not frame-by-frame).
- Count an object as moving if any of the following holds:
  (a) Translation: position changes relative to the scene/background (parallax-aware).
  (b) Rotation/orientation change: sustained rotation or turning (e.g., door opening, car turning).
  (c) Articulation or deformation driven by the object or its operator (e.g., walking, running, jumping, swinging, throwing, opening, closing, lifting, pushing, pulling, rolling, bouncing, striking, strumming, typing, pedaling).
- Do not count as moving: motion caused only by camera pan/zoom/tilt/shake; microscopic flicker or aliasing; motion that is ambiguous or below perceptual threshold.

Format:
- Keys are short lowercase noun phrases naming moving objects; include simple attributes for disambiguation (e.g., `runner in striped shirt`, `red car`).
- Values are short gerund verb phrases (1–3 words), optionally with a direct object (e.g., `drifting`, `pushing cart`, `opening door`).

Objects to exclude as keys:
- Continuous media: water (ocean, river, waves), atmospheric effects (smoke, fog, clouds), fire, precipitation (rain, snow, hail), wind/storms, volcanic material (lava, ash).
- Clothing/accessories as standalone objects (shirts, hats, logos, patterns); these may appear only as attributes to describe an entity.

Output format:
```
{"object":  "state/action", ...}
```

Examples:
```
{"red car":"drifting","black pickup truck":"colliding"}
{"basketball":"bouncing","player":"running"}
{"door":"opening","person":"pushing"}
{"runner in striped shirt":"sprinting","black dog":"chasing"}
{"shopping cart":"rolling","man":"pushing"}
```

Table S8. **Prompt used for mass estimation in the Physics Parameter Estimator (PPE).**

You will be shown a short video. From this video, moving objects and their visible state/action are provided as a JSON map, `MOVING OBJECTS: {"object": "state/action", ...}`

Task:
For every key in `MOVING OBJECTS`, estimate its mass in kilograms and return exactly one JSON object of the form:
`{"object":{"rationale":"<concise explanation ending with the same number>","weight_kg":<number>}, ...}`

Policy:
- Use only the keys given in `MOVING OBJECTS`; do not invent, rename, split, or merge keys.
- Include every key exactly once, in the same order as in `MOVING OBJECTS`.
- Keys must match `MOVING OBJECTS` verbatim, character-for-character.
- `"weight_kg"` must always be a numeric value (no units, no text).
- Rationales should be concise, reference visible attributes (e.g., material, dimensions, body build), and end with the same number reported in `"weight_kg"`.
- Avoid round anchor numbers unless clearly justified.
- Base estimates on visible material, thickness, density, and overall volume in the video. For standardized items use specs (convert lb→kg by ÷2.20462).
- Return only the final JSON object. No markdown, no code fences, no extra text.

Examples:
```
MOVING OBJECTS: {"black pickup truck":"colliding","red car":"drifting"}
OUTPUT: {"black pickup truck":{"rationale":"Mid-size double-cab pickup;
steel frame → about 1925.0 kg","weight_kg":1925.0},
       "red car":{"rationale":"Compact sedan; 4-door body → about 1410.0
kg","weight_kg":1410.0}}
MOVING OBJECTS: {"basketball":"bouncing","player":"running"}
OUTPUT: {"basketball":{"rationale":"Official size 7 ball; inflated rubber
bladder → about 0.62 kg","weight_kg":0.62},
       "player":{"rationale":"Adult male; medium build → about 72.0
kg","weight_kg":72.0}}
MOVING OBJECTS: {"dog":"jumping","suitcase":"rolling"}
OUTPUT: {"dog":{"rationale":"Medium-size Labrador; lean build → about 27.0
kg","weight_kg":27.0},
       "suitcase":{"rationale":"Hard-shell carry-on, empty → about 3.8
kg","weight_kg":3.8}}
MOVING OBJECTS: {"guitar":"being strummed","chair":"falling"}
OUTPUT: {"guitar":{"rationale":"Full-size acoustic guitar with wooden body
→ about 2.2 kg","weight_kg":2.2},
       "chair":{"rationale":"Plastic molded chair, lightweight → about 3.5
kg","weight_kg":3.5}}
MOVING OBJECTS: {"quadcopter drone":"hovering","bicycle":"coasting"}
OUTPUT: {"quadcopter drone":{"rationale":"Prosumer quadcopter with gimbal;
typical takeoff mass → about 0.90 kg","weight_kg":0.90},
       "bicycle":{"rationale":"Aluminum road bike, no racks/fenders → about
9.5 kg","weight_kg":9.5}}
```

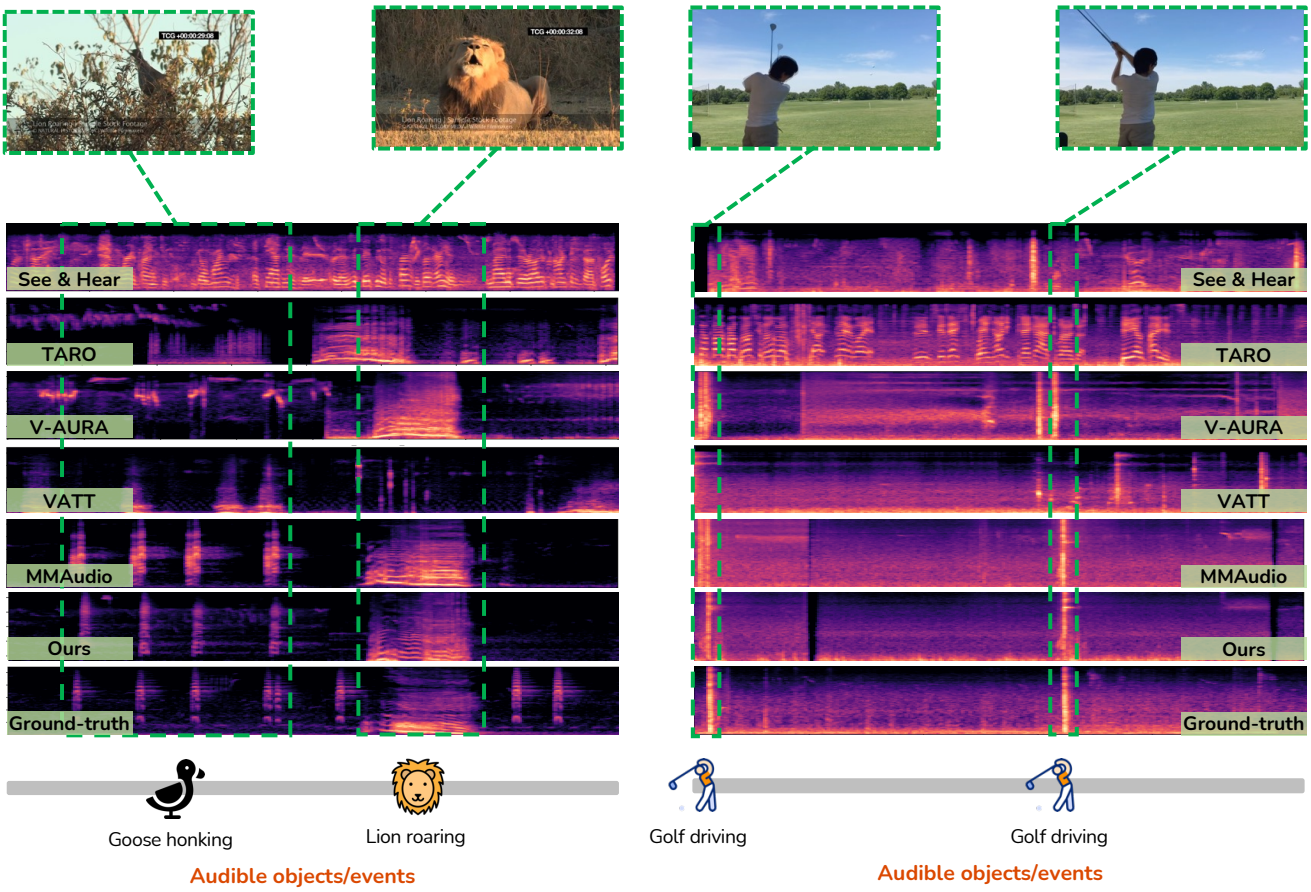

Figure S5. **Qualitative comparison of generated spectrograms.** We present spectrogram visualizations from the state-of-the-art video-to-audio generation models [6, 14, 18, 19, 24], ours, and the ground-truth audio on VGGSound test split. Green dashed lines highlight spectral structures that correspond to visually observable events, and icon markers indicate audible objects or interactions. **[Left]** PAVAS produces spectral patterns that follow the timing and duration of the visual events more faithfully. For example, during the goose honking moment, our model avoids generating unnaturally prolonged honks—a pattern observed in MMAudio [6]. **[Right]** In the golf driving example, V-AURA [19] and MMAudio [6] incorrectly generate *two* impact sounds at the beginning of the sequence. In contrast, ours produces a single, sharply localized impact that matches the ground-truth spectrogram.

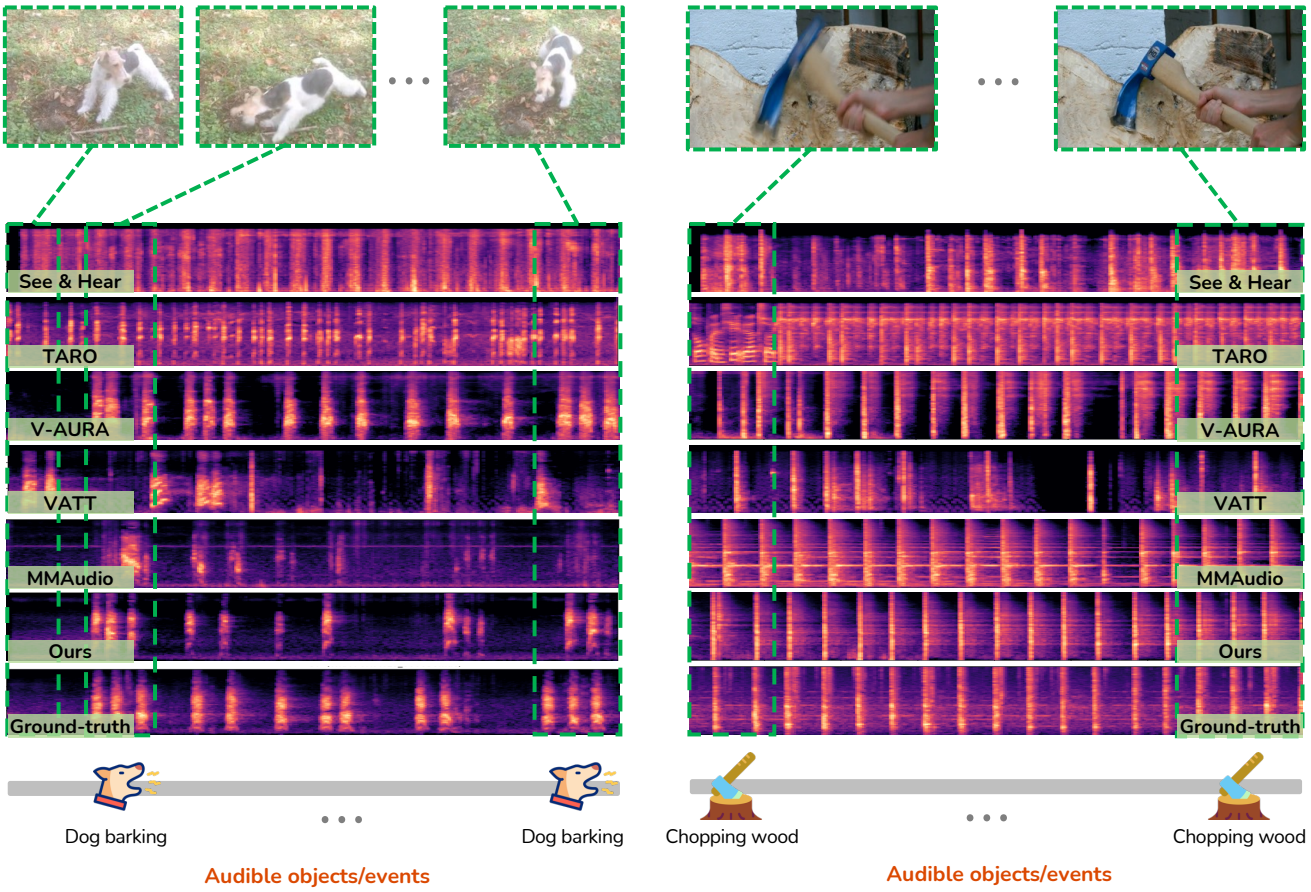

Figure S6. **Qualitative comparison of generated spectrograms.** We present spectrograms produced by state-of-the-art video-to-audio models [6, 14, 18, 19, 24], PAVAS, and the ground truth. Green dashed lines mark spectral patterns that correspond to visual events, and icon markers denote audible objects or interactions in the scene. **[Left]** In the dog-barking example, PAVAS most accurately reproduces the timing and spectral shape of the bark. Competing models either miss the barking moment or fail to emit a bark at all when the dog opens its mouth. **[Right]** In the wood-chopping scene, PAVAS generates impact transients that are temporally aligned with the chopping motion and exhibit spectral structures closely matching the ground truth. Other models tend to produce misaligned strikes or spectrogram patterns that deviate noticeably from the sharp, short-decay broadband signature of real impact sounds.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1, 4, 5

[2] Sebastian Böck and Gerhard Widmer. Maximum filter vibrato suppression for onset detection. In *the 16th International Conference on Digital Audio Effects (DAFx-13)*, 2013. 3

[3] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*, 2024. 2

[4] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset. In *ICASSP*, 2020. 1

[5] Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 5

[6] Ho Kei Cheng, Masato Ishii, Akio Hayakawa, Takashi Shibuya, Alexander Schwing, and Yuki Mitsufuji. Mmaudio: Taming multimodal joint training for high-quality video-to-audio synthesis. In *CVPR*, 2025. 1, 2, 3, 8, 9, 12, 13

[7] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *CVPR*, 2022. 1

[8] Vladimir Iashin, Weidi Xie, Esa Rahtu, and Andrew Zisserman. Synchformer: Efficient synchronization from sparse cues. In *ICASSP*, 2024. 6

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[10] Lawrence E Kinsler, Austin R Frey, Alan B Coppens, and James V Sanders. *Fundamentals of acoustics*. John wiley & sons, 2000. 3

[11] Sang-gil Lee, Wei Ping, Boris Ginsburg, Bryan Catanzaro, and Sungroh Yoon. Bigvgan: A universal neural vocoder with large-scale training. In *ICLR*, 2023. 6

[12] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932. 9

[13] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 1, 2

[14] Xiulong Liu, Kun Su, and Eli Shlizerman. Tell what you hear from what you see-video to audio generation through text. In *NeurIPS*, 2024. 8, 9, 12, 13

[15] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2, 4, 5, 8

[16] Kazuki Shimada, Archontis Politis, Parthasaarathy Sudarsanam, Daniel A Krause, Kengo Uchida, Sharath Adavanne, Aapo Hakala, Yuichiro Koyama, Naoya Takahashi, Shusuke Takahashi, et al. Starss23: An audio-visual dataset of spatial recordings of real scenes with spatiotemporal annotations of sound events. In *NeurIPS*, 2023. 1

[17] Trevor Standley, Ozan Sener, Dawn Chen, and Silvio Savarese. image2mass: Estimating the mass of an object from its image. In *Conference on Robot Learning*, 2017. 1, 2

[18] Tri Ton, Ji Woo Hong, and Chang D Yoo. Taro: Timestep-adaptive representation alignment with onset-aware conditioning for synchronized video-to-audio synthesis. In *ICCV*, 2025. 1, 8, 9, 12, 13

[19] Ilpo Viertola, Vladimir Iashin, and Esa Rahtu. Temporally aligned audio for video with autoregression. In *ICASSP*, 2025. 8, 9, 12, 13

[20] Heng Wang, Jianbo Ma, Santiago Pascual, Richard Cartwright, and Weidong Cai. V2a-mapper: A lightweight solution for vision-to-audio generation by connecting foundation models. In *AAAI*, 2024. 8

[21] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *CVPR*, 2025. 4, 5, 8

[22] Eric W. Weisstein. Gnomonic projection. MathWorld–A Wolfram Resource, 2025. 2

[23] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *CVPR*, 2024. 4, 5

[24] Yazhou Xing, Yingqing He, Zeyue Tian, Xintao Wang, and Qifeng Chen. Seeing and hearing: Open-domain visual-audio generation with diffusion latent aligners. In *CVPR*, 2024. 8, 9, 12, 13

[25] Albert J Zhai, Yuan Shen, Emily Y Chen, Gloria X Wang, Xinlei Wang, Sheng Wang, Kaiyu Guan, and Shenlong Wang. Physical property understanding from language-embedded feature fields. In *CVPR*, 2024. 1, 2