APPENDIX: TRAINING DETAILS AND REPRODUCIBILITY

A.1 REPRODUCIBILITY

Anonymized code and demo datasets will be available on our webpage (`https://anonymous1415510-spec.github.io`). We provide details about comparison with other algorithms to facilitate the reproducing of our results. All details about the hyperparameters, environment specifications, and real-world experiment setup are provided in the appendix or the website.

A.2 MODEL ARCHITECTURE AND RENDERER DETAILS

We adopt the `AVRModel_complex_FD_FreqDep_PhaseCorrection` model architecture with the `AVRRenderFD_FreqDep_PhaseCorrection_KK` renderer. This setup is designed for complex-valued frequency domain rendering and enables physically grounded learning with explicit modeling of attenuation and phase velocity.

Key architectural components:

- **Complex Frequency Field Prediction**: The model learns frequency-dependent attenuation fields $\delta(f) = \sigma(f) + j\beta(f)$ and complex responses $H(f)$ using MLPs operating on hash-encoded spatial coordinates.
- **Separate Signal and Attenuation Networks**: Frequency-specific signal and attenuation values are predicted using distinct encoders and MLPs.
- **Directional Encodings**: Transmitter and receiver directions are encoded using spherical harmonics.
- **Renderer Pipeline**: Rays are sampled in spherical directions with integration over 64 azimuth × 32 elevation rays, each with 64 samples from near=0 to far=4 meters. Cumulative attenuation is applied using $\exp(-\sum \sigma_i \Delta u_i + j \sum \beta_i \Delta u_i)$.

Parameter	Value
Learning Rate	5×10^{-4} (cosine annealing to 5×10^{-5})
Optimizer	Adam
Total Iterations	15,000
Batch Size	1
Rendering Samples	64 per ray
Azimuth × Elevation Rays	64×32
Speed of Sound	343.8 m/s
Sampling Frequency	48,000 Hz
Path Loss Exponent	1
Layers	8 fully connected layers
Hidden Units	256 neurons per layer
Activation	ReLU
Positional Encoding	10 frequencies for spatial and directional input

Table 1: Training hyperparameters and network architecture for INFER.

A.3 LOSS FUNCTIONS AND WEIGHTS

The total training objective is composed of multiple terms designed to supervise the model's output across spectral amplitude, phase, energy structure, and physical consistency. The overall loss is expressed as:

$$L_{\text{total}} = \lambda_{\text{spec}} L_{\text{spec}} + \lambda_{\text{mag}} L_{\text{mag}} + \lambda_{\text{phase}} L_{\text{phase}} + \lambda_{\text{env}} L_{\text{env}} + \lambda_{\text{energy}} L_{\text{energy}} + \lambda_{\text{KK}} L_{\text{KK}} + \lambda_{\text{STFT}} L_{\text{MR-STFT}}. \tag{1}$$

In our experiments, we set $\lambda_{\text{spec}} = 16$, $\lambda_{\text{mag}} = 4$, $\lambda_{\text{phase}} = 1$, $\lambda_{\text{env}} = 0.25$, $\lambda_{\text{energy}} = 2$, $\lambda_{\text{KK}} = 0.25$, and $\lambda_{\text{STFT}} = 0.25$. Beyond these global weights, we apply frequency-dependent weighting: for

L_{phase} we emphasize low and mid frequencies by setting $w(f) = 1.2$ up to 1.5 kHz (125 bins), $w(f) = 1$ until 5 kHz (425 bins), and then smoothly tapering to 0.8 across the log-frequency axis; for L_{mag}, weights are 1 up to 1.5 kHz, 1.25 until 5 kHz, and then tapered to 0.8; and for L_{spec}, we assign 1.25 up to 5 kHz before tapering. These schedules follow psychoacoustic informed weighting strategies, prioritizing perceptually critical bands while deemphasizing unreliable high-frequency bins.

A.4 Training Pipeline

We use the script `avr_runner_complex_FD_kk.py` for training. Each step involves:

1. Ray-based spherical integration using normalized receiver and transmitter coordinates.
2. Prediction of complex signals and attenuation fields.
3. Loss computation including frequency-weighted spectrum and KK regularization.
4. Gradient backpropagation with NaN filtering and norm clipping.
5. Mixed precision training and GPU memory optimization.

Special Considerations:

- Gradient clipping to max norm 1.
- Automatic mixed precision (AMP) to save memory.
- Complex loss handling via separate $\Re[H(f)]$ and $\Im[H(f)]$ paths.
- KK regularizer computed using discrete Hilbert transform with frequency masking and tapering.

A.5 Reproducibility Checklist

Software Environment:

- `Python 3.8`, `PyTorch 1.12.0`, `CUDA 11.6`
- `tinycudann 1.6`, `auraloss 0.4.0`, `tensorboard 2.8.0`

Training Script (Single GPU):

```
python avr_runner_complex_FD_kk.py \
  --config config_files/avr_buck_complex_dir_FD.yml \
  --model_type AVRModel_complex_FD_FreqDep_PhaseCorrection \
  --renderer_type AVRRenderFD_FreqDep_PhaseCorrection_KK \
  --batchsize 1 \
  --dataset_dir /path/to/dataset
```

A.6 Hardware Requirements

Minimum:

- GPU: NVIDIA RTXA6000 (10GB+ VRAM)

Recommended:

- GPU: L40S

Training time is approximately 24 hours on a single L40S.

A.7 Evaluation Metrics

- **Spectral Distance (SD)**: $20 \log_{10}(\||H(f)| - |\hat{H}(f)|\|/\||\hat{H}(f)|\|)$
- **Phase Error (PRE)**: $\|\angle H(f) - \angle \hat{H}(f)\|_2/\pi$
- **Energy Ratio, Cumulative Energy Error** in frequency domain
- **Time-Domain Metrics**: PSNR, MR-STFT loss
- **KK Violation Metric**: $\|\beta(f) - \mathcal{H}[\sigma(f)]\|/\|\beta(f)\|$

A.8 Audio Hardware Specification

We detail here the acoustic transducer setup used for our data collection in the *BUCK* testbed and the production *Tesla Model X* vehicle. Both systems were equipped with a rich spatial arrangement of loudspeakers and a high-fidelity microphone array to facilitate spatial audio capture and reconstruction.

Speaker Configuration. While the Tesla Model X uses the default speakers, In BUCK, the active speakers used for sound excitation include:

- **Center Dash Speaker:** A 3.5-inch wideband driver, such as the SLA Ram3 or Dayton Audio DMA90-4, capable of full-range output from $85.00\,\mathrm{Hz}$ to $20.00\,\mathrm{kHz}$. In typical configurations, these are high-passed at approximately $100.00\,\mathrm{Hz}$ to avoid low-frequency distortion.
- **Rear Door Speakers:** Morel Tempo Ultra Integra 402 or 602 coaxial hybrids with wideband support ($55.00\,\mathrm{Hz}$ to $22.00\,\mathrm{kHz}$), high sensitivity, and power handling up to $120.00\,\mathrm{WRMS}$. These speakers internally crossover between woofer and tweeter around $2.50\,\mathrm{kHz}$–$3.00\,\mathrm{kHz}$.
- **Rear Height Speakers:** Tang Band T2-2136SF full-range modules and Morel CCWR254 midrange drivers, mounted in ceiling/rear hatch positions to introduce vertical spatial content, spanning $80.00\,\mathrm{Hz}$ to $20.00\,\mathrm{kHz}$. Crossover filters are typically applied around $800.00\,\mathrm{Hz}$–$1.00\,\mathrm{kHz}$ depending on system design and companion driver.

This layout approximates a $7.1.4$ immersive audio setup and enables extensive sampling of reverberant and directional field responses across both testbeds.

Microphone Array. We use the commercially available **MiniDSP UMA-16** USB microphone array, which offers 16 omnidirectional MEMS microphones in a linear array form factor. This array supports high-resolution spatial sampling across the cabin, enabling dense reconstruction of directional impulse responses.

A.9 Baseline Methods

To rigorously evaluate the effectiveness of our proposed system INFER, we compare against three representative baselines, each reflecting a different class of acoustic modeling approach:

- **AVR**: A hybrid time–frequency domain neural field that learns time-domain impulse responses via a differentiable renderer. While AVR applies frequency-domain path delays in its rendering, the model is supervised in the time domain and does not explicitly learn frequency-dependent attenuation or dispersion.
- **NAF (Neural Acoustic Field)**: A neural field trained directly in the time domain using MSE and time-domain perceptual losses. NAF ignores frequency-domain supervision and is evaluated primarily on time-domain waveform fidelity.
- **INRAS (Impulse Response as Signal)**: A signal regression approach where the model directly regresses to the complex impulse response waveform as a 1D signal. INRAS uses STFT-based perceptual loss, but it does not exploit any spatial priors or directional conditioning.

Each baseline is re-implemented in our codebase with their respective loss functions and evaluation metrics faithfully reproduced, using the same training datasets, preprocessing pipelines, and neural architecture backbones where applicable.

A.10 Training Configuration

All baselines are trained with identical configurations to ensure fair comparisons:

- **Dataset:** We use the same training/validation/test splits from our real (BUCK) and synthetic (Tesla) datasets for all methods.
- **Resolution:** The frequency bins, spatial sampling resolution, and directional integration are matched across all methods.
- **Training Epochs:** All models are trained for 500 epochs with early stopping based on validation loss.
- **Batch Size:** Batch size of 1 is used due to GPU memory constraints, consistent with prior volumetric rendering works.
- **Evaluation:** All comparisons are evaluated on both magnitude and phase accuracy across the frequency range of interest, in addition to perceptual STFT loss and energy-based metrics.

A.11 Loss Function Implementation

AVR. We follow the original AVR formulation and use the same set of losses described in the paper.

NAF. The NAF baseline is trained using the standard losses introduced in its original work, without any additional frequency-domain regularization.

INRAS. For INRAS, we adopt the exact losses specified in the original paper, without modification.

A.12 Architectural Modifications

To isolate the effects of spectral supervision and renderer formulation, all baseline models are built upon the same backbone MLP architecture as our method:

- 8-layer fully connected network with sinusoidal positional encoding.
- Input: $(\mathbf{p}_{\text{tx}}, \hat{\mathbf{n}}_{\text{tx}}, \mathbf{x}, \hat{\mathbf{n}})$ with appropriate frequency and spatial encodings.
- Output: Real-valued waveform or complex spectrum depending on method.

A.13 Evaluation Metrics

We evaluate all baselines and our method using the following metrics:

- **Amplitude Error**: L1 distance between predicted and ground-truth magnitude spectra across frequency.
- **Phase Error**: Cosine-sine angular difference between predicted and true phase.
- **Envelope Error**: Smoothed log-magnitude deviation (via exponential smoothing).
- **MR-STFT Loss**: Multi-resolution STFT loss to evaluate perceptual reconstruction in time domain.
- **Energy Accuracy**: Deviation from cumulative spectral energy profiles.

A.14 Notes on Fairness and Robustness

To ensure fairness in evaluation:

- All models are trained with the same GPU hardware, random seed initialization, and PyTorch version.
- We use the same optimizer (Adam) and learning rate schedule across all models unless otherwise noted.
- All baselines are evaluated using our standardized renderer and metric pipeline to eliminate post-processing inconsistencies.
- We tune loss weights and learning rates for each baseline to ensure their best performance under our training conditions.

Overall, our comparison demonstrates that INFER substantially outperforms these baselines across spectral and perceptual metrics due to its physics-informed supervision and frequency-aware modeling.