Real Acoustic Fields: An Audio-Visual Room Acoustics Dataset and Benchmark

Supplementary Material

A.1. Additional Experimental Results

Benchmark on 16 KHz impulse responses. We also evaluate each method on our benchmark with impulse responses of 16 kHz sampling rate. We show the results in Table 7. We can see that INRAS++ performs best overall, which matches with the conclusion in Section 5.1.

More qualitative results. We provide more predicted RIR visualization for qualitative comparison in Figure 9. We also provide more loudness map visualization on the different scenes for qualitative comparison in Figure 10.

Empty versus furnished room. One advantage of our dataset is that it contains a scene in two conditions — empty and furnished, which allows studying the difference in acoustic fields introduced by furniture. Due to a lack of ground-truth comparison, we visualize the generated impulse responses from INRAS++ trained on each scene individually as an approximation of the acoustic field. We show our results in Figure 7, where we can see that generated impulse responses with different acoustic properties.

A.2. Implementation Details

In this section, we will demonstrate the implementation of each baseline in detail.

AAC and Opus. We convert the raw waveform (`.wav`) into AAC (`.m4a`) and Opus (`.opus`) encoding and reverse the compression using FFmpeg commands as shown below:

```
# AAC compression
encode_command = f"ffmpeg -i audio.wav -t {
    audio_length} -c:a aac -b:a 24k temp.m4a"
decode_command = f"ffmpeg -i temp.m4a -c:a
    pcm_f32le -ar {sampling_rate} audio_aac.wav"

# Opus compression
encode_command = f"ffmpeg -i audio.wav -t {
    audio_length} -c:a opus -strict -2 -b:a 24k
    temp.opus"
decode_command = f"ffmpeg -i temp.opus -c:a
    pcm_f32le -ar {sampling_rate} audio_opus.wav"
```

Listing 1. FFmpeg commands for audio compression

We cut the audio to be the same length (0.32s) and corresponding sampling rate (16K or 48K) for fair evaluation comparison.

Note that we use a different Opus encoder which can achieve better compression performance than NAF used [36]. Due to the heavy computation of constructing a high-dimensional interpolation engine, we modify the baseline algorithm by first matching the nearest neighbor of the emitter in the training distribution and then performing the nearest neighbor or linear interpolation to generate impulse responses for given listener positions.

NAF. We follow the official implementation of NAF [2], and create 3D grid features based on the bounding boxes of scenes. For experiments with 16 kHz sampling rate, we use an STFT with an FFT size of 512, a window length of 256, and a hop length of 128. For 48 kHz sampling rate, we use an STFT with an FFT size of 1024, a window length of 512, and a hop length of 256. We perform inverse STFT on the predicted magnitude of the RIR spectrogram with random spectrogram phase to obtain time-domain RIR. We set $\lambda = 1.0$ for the weight of energy decay loss when training NAF++.

INRAS. We follow the implementation of INRAS provided by the authors in their supplementary material[3], and add an extra dimension for the emitter, listener, and bounce point position. We changed the original bounce point sampling method, which only sampled points with a specific height. Instead, we apply Poisson sampling on the scene meshes to obtain 256 bounce points in 3D to represent scene geometry in a better way. To optimize multi-resolution STFT loss, we set FFT size as $\{128, 512, 1024, 2048\}$, window length as $\{80, 240, 600, 1200\}$, and hop length as $\{16, 50, 120, 240\}$. We set $\lambda = 2.0$ for the weight of the energy decay loss.

NACF. We use the same architecture as INRAS for NACF. We keep the original bounce point sampling strategy in the paper and render visual context using VR-NeRF [66]. We render 256×256 pixel RGB, and depth images with a field of view of 90°. We use the surface normal of each bounce point to determine the look-at view of the virtual camera. Following the original paper, RGB and depth images are down-sampled to 16×16 and are encoded with an MLP as visual contexts. We set $\lambda = 2.0$ for the weight of energy decay loss. We optimize the multi-resolution STFT loss with the same hyperparameters as INRAS.

AV-NeRF. Because we have a different setup from AV-NeRF [33] where we have omnidirectional microphones instead of orientated binaural receivers, we adopt their method with several changes. We use VR-NeRF [66] to render 4 perspective views of 256×256 RGB and depth maps with a field of view of 90° for each receiver's position, and encode them with frozen ResNet18 [23] trained on ImageNet [17]. We removed the relative angle because it does not fit our setup. We set $\lambda = 2.0$ for the weight of energy decay loss.

[2] https://github.com/aluo-x/Learning_Neural_Acoustic_Fields/
[3] https://openreview.net/forum?id=7KBzV5IL7W

Table 7. **Benchmark with 16 kHz sampling rate.**

	Method	Variation	STFT error (dB) ↓	C_{50} error (dB) ↓	EDT error (sec) ↓	T_{60} error (%) ↓	Parameters (Million) ↓	Storage (MB) ↓	Speed (ms) ↓
Classical	Linear	AAC	1.14	1.09	0.040	8.79	–	680.45	–
		Opus	1.06	0.80	0.032	7.48		680.45	
		original	1.02	0.82	0.032	6.82		3,172.77	
	Nearest	AAC	0.72	0.83	0.027	8.08	–	680.45	–
		Opus	0.58	0.61	0.020	6.96		680.45	
		original	0.48	0.71	0.020	7.68		3,172.77	
Neural	NAF [36]	vanilla	0.77	0.69	0.025	8.15	5.51	22.04	5.57
		+ decay loss	0.77	0.63	0.023	7.43			
	INRAS [57]	vanilla	**0.44**	0.65	0.024	6.15	**1.33**	**5.31**	**2.10**
		+ decay loss	0.45	**0.54**	**0.019**	**5.34**			
	NACF [34]	vanilla	0.45	0.58	0.020	5.47	1.52	6.05	2.39
		+ temporal	0.48	0.60	0.022	6.59	1.75	7.00	2.78
	AV-NeRF [33]	vanilla	0.46	0.58	0.021	6.12	12.99	51.98	5.80

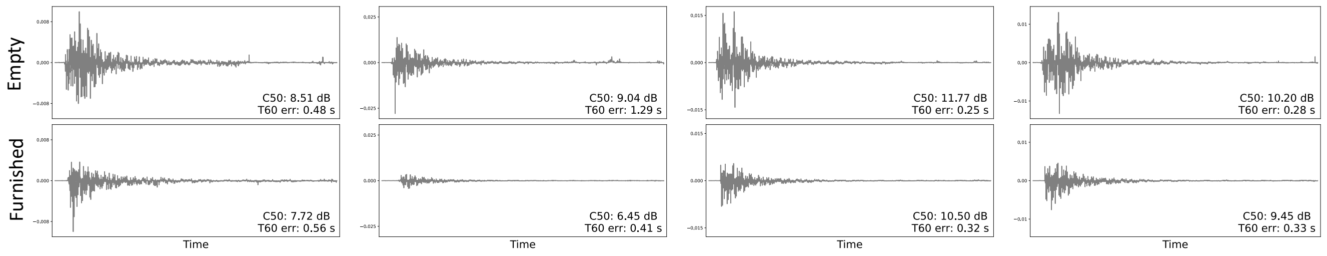

Figure 7. **Visualization comparison of generated RIRs from different scenes.** We present visualizations of four pairs of generated impulse responses, each sharing the same emitter-receiver position in both the empty room and the furnished room. These visualizations highlight the variations in the acoustic fields between the two distinct scenes.

A.3. Dataset

Impulse response data processing. We followed the sine-sweep deconvolution process as described by Farina [18] to extract the impulse response from the signals recorded by the microphones. For each extracted impulse response, we saved the 3D location of the receiver, as well as the 3D location and orientation of the sound source. The length of the impulse response is 4 seconds and all audio data was recorded at a sampling rate of 48 kHz and stored at a resolution of 32 bits. We show the RT60 distribution of our collected RIRs in Figure 8

Visual rendering. We provide renderings of room meshes as a simple overview in Figure 11.

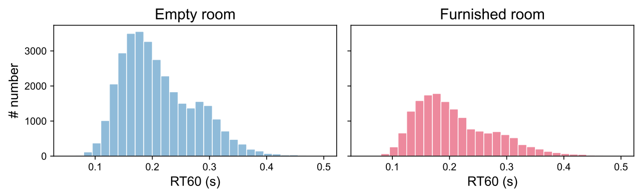

Figure 8. **RT60 distribution.**

Speaker orientation. In Figure 12, we provide visualizations of impulse response pairs from our captured dataset. These pairs share the same emitter-listener position but differ in emitter orientations. The orientations of directional speakers impact the resulting impulse responses.

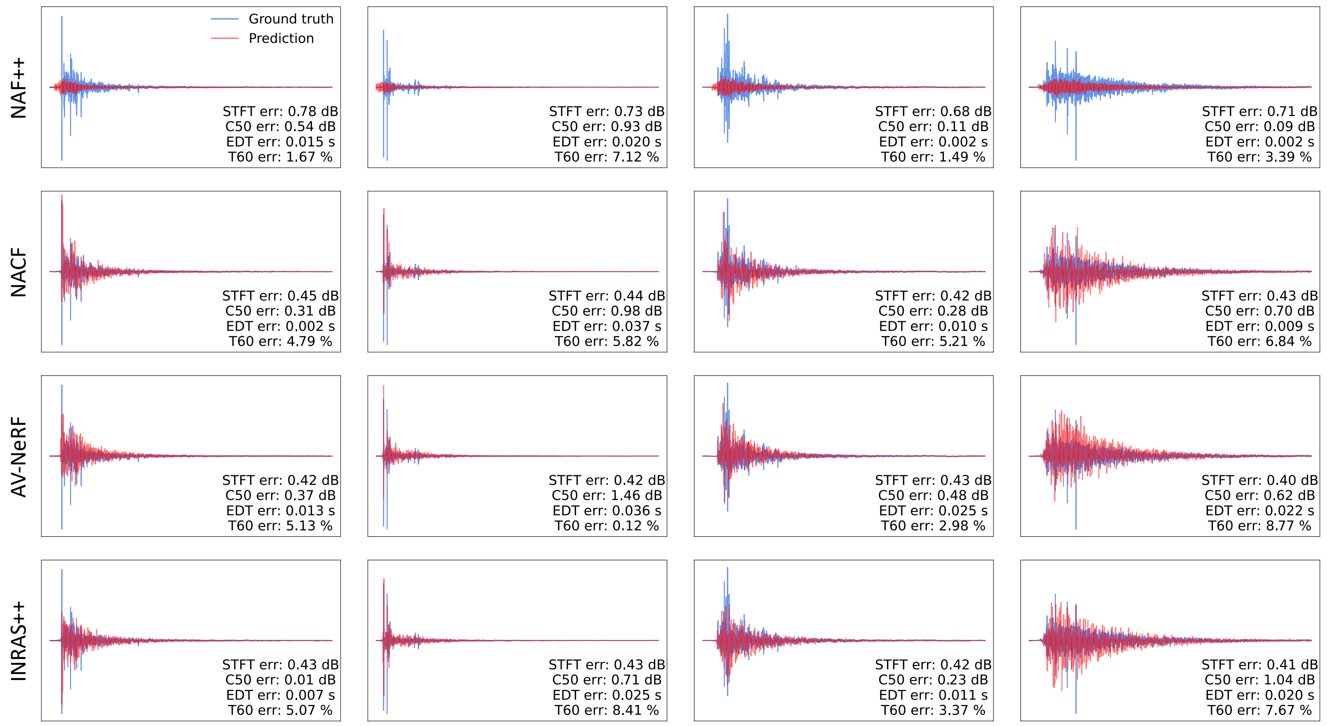

Figure 9. **Visualization of generated RIRs.** We visualize the ground truth (in blue) and predicted (in red) impulse responses of several methods for qualitative comparison.

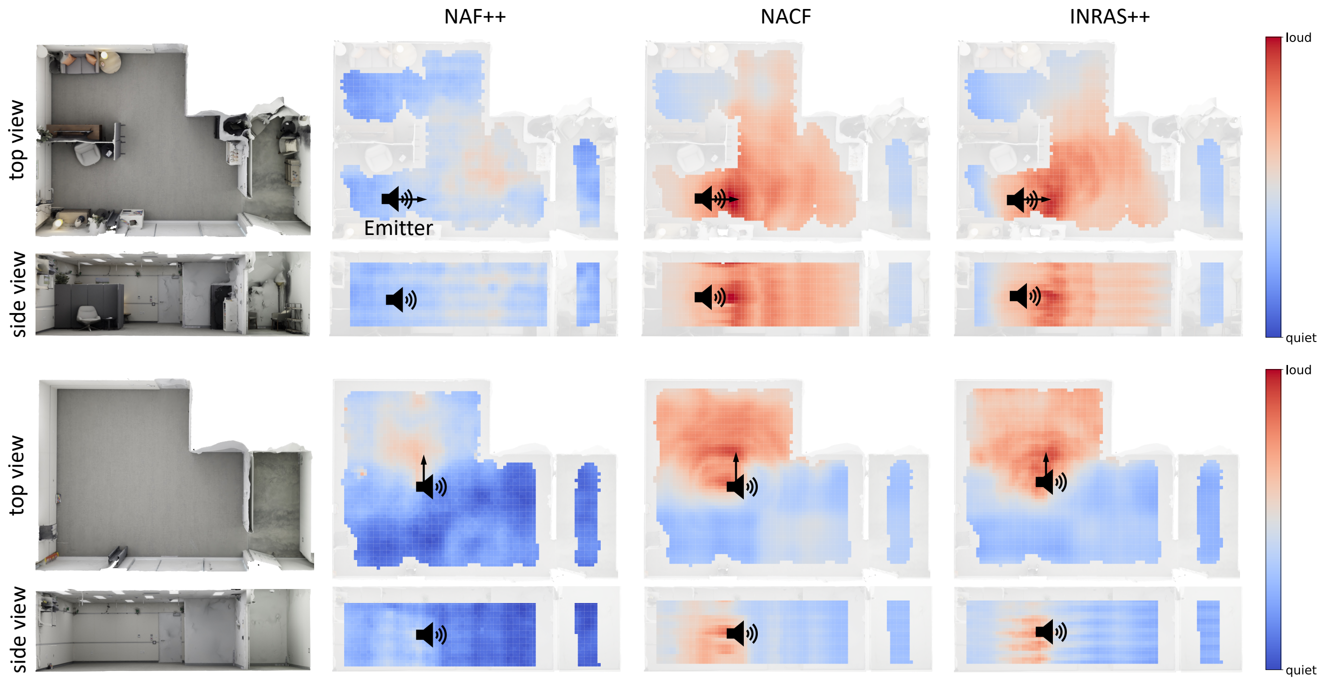

Figure 10. **Loudness map visualization.** We visualize the intensity of predicted impulse responses over the spaces from the top view and side view given an emitter position and its orientation. Red means loud and blue means quiet.

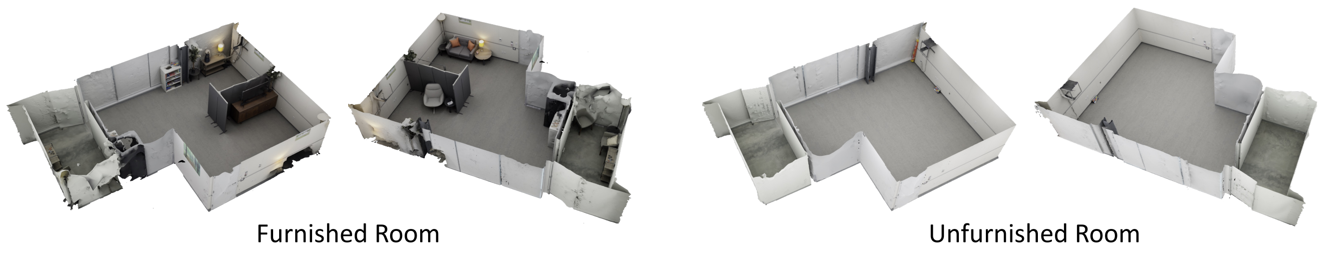

Figure 11. **Scene overview of RAF.**

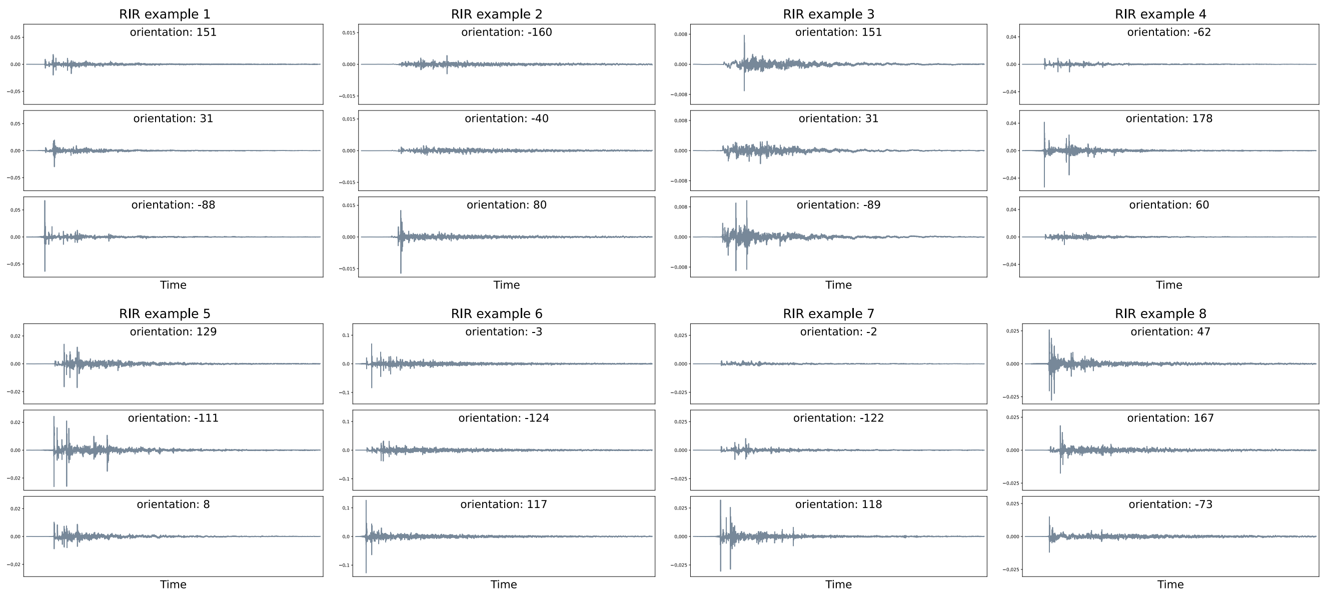

Figure 12. **Visualization of ground-truth RIRs with different orientations.**