Appendix A Supplementary material

In this supplementary material we provide additional details about:

- Video (with audio) for qualitative illustration of our task and qualitative evaluation of our model predictions (Sec. A.1)
- Audio dataset details (Sec. A.2), as mentioned in Sec. 5 of the main paper
- Model architecture details for RIR prediction (Sec. A.3.1), downstream tasks (Sec. A.3.2) and our audio-visual policy (Sec. A.4), as noted in Sec. 5 of the main paper

A.1 Supplementary Video

The supplementary video shows the perceptually realistic SoundSpaces [13, 30] audio simulation platform that we use for our experiments, and provides a qualitative illustration of both the *passive* and *active sampling* variants of our task, Sample-efficient Audio-Visual Learning of Scene Acoustics. Moreover, for passive sampling, we qualitatively demonstrate our model's prediction quality by comparing the predictions with the ground truths, both at the RIR level and in terms of perceptual similarity when the RIRs are convolved with real-world monaural sounds, like speech and music. We also analyze common failure cases for our model for this variant and qualitatively show how our model predictions can be used to successfully localize an audio source in a 3D environment. For active sampling, we first qualitatively model and reward design, then show qualitative results highlighting the agent's trajectories as it tries to gather acoustically informative samples. Please use headphones to hear the spatial audio correctly. The video is available at http://vision.cs.utexas.edu/projects/fewShot-RIR.

A.2 Audio Dataset

For our experiment with ambient environment sounds (Sec. 5.2.1 in main), we use ambient sounds from the ESC-50 [101] dataset (e.g., dog barking, running water). For every test query, we randomly sample a location in the 3D scene for an ambient sound and play a randomly chosen 1 second long clip from the ESC-50 [101] dataset at that location. To retrieve the observed binaural echo response A_i (Sec. 3 and 4.1.1 in main) in this setting, we first convolve the clean echo RIR for each observation O_i with the sinusoidal sweep sound, then mix it with the binaural the ambient sound for its pose P_i, and finally deconvolve using the inverse sweep (Sec. 4.1.1 in main).

A.3 Model architecture

Here, we provide our architecture and additional training details for reproducibility. We publish the code for this work at:
http://vision.cs.utexas.edu/projects/fs_rir
https://vision.cs.utexas.edu/projects/active_rir/

A.3.1 Model architectures for RIR prediction

Visual encoder. Our visual encoder f^V is a ResNet-18 [86] model (Sec. 4.1.1 in main) that takes egocentric RGB and depth images from the observation set, which are concatenated channel-wise, as input and produces a 512-dimensional feature.

Acoustic encoder. Our acoustic encoder f^A is another ResNet-18 [86] (Sec. 4.1.1 in main) that separately encodes the binaural log magnitude spectrogram for an echo RIR A_i into a 512-dimensional feature.

Pose encoder. To embed an observation pose P_i or a query source-receiver pair Q (Sec. 3 in main), we use sinusoidal positional encodings [85] (Sec. 4.1.1 in main) with 8 frequencies, which generate a 16-dimensional feature vector (the positional encodings comprise both sine and cosine components with 8 features per component) for every attribute of an observation pose or a query (i.e., x, y, and θ).

Modality encoder. For our modality embedding m (Sec. 4.1.1 in main), we maintain a sparse lookup table of 8-dimensional learnable embeddings, which we index with 0 to retrieve the visual modality embedding (m_V) and 1 to retrieve the acoustic modality embedding (m_A).

Fusion layer. To generate the multimodal memory S (Sec. 4.1.1 in main) for our context encoder (Sec. 4.1.1 in main), we separately concatenate the modality features (produced by f^V for vision and f^A for echo responses) for an observation, the corresponding sinusoidal pose embedding, and the modality embedding (m_V for visual features and m_A for acoustic features), and project using a single linear layer to 1024-dimensional embedding space. Similarly, to generate the query encoding q for our conditional RIR predictor (Sec. 4.1.2 in main), we use another linear layer to project the query's sinusoidal positional encodings to a 1024-dimensional feature vector. Furthermore, we don't use bias in any fusion layer.

Context encoder. Our context encoder (Sec. 4.1.1 in main) is a transformer encoder [85] with 6 layers, 8 attention heads, a hidden size of 2048 and ReLU [102, 103] activations. Additionally, we use a dropout [104] of 0.1 in our context encoder.

Conditional RIR predictor. Our conditional RIR predictor (Sec. 4.1.2 in main) has 2 components: 1) a transformer decoder [85] to perform cross-attention on the implicit representation C (Sec. 4.1.1 in main), which is produced by the previously described context encoder, using the query encoding q (Sec. 4.1.2 in main), and 2) a multi-layer transpose convolution network U (Sec. 4.1.2 in main) to upsample the decoder output d^Q and predict the magnitude spectrogram for the query Q in log space.

The transformer decoder [85] has the same architecture as our context encoder.

The transpose convolution network U comprises 7 layers in total. The first 6 layers are transpose convolutions with a kernel size of 4, stride of 2, input padding of 1, ReLU [102, 103] activations and BatchNorm [105]. The number of input channels for the transpose convolutions are 128, 512, 256, 128, 64 and 32, respectively. The last layer of U is a convolution layer with a kernel size of 3, stride of 1, padding of 1 along the height dimension and 2 along the width dimension, and 16 input channels. Finally, we switch off bias in all layers of U.

A.3.2 Model architectures for downstream tasks

Sound source localization. We use a ResNet-18 [86] feature encoder that takes the log magnitude spectrogram of an RIR (predicted or ground truth as input). We take the encoded features and feed them to a single linear layer that predicts the location coordinates of a query's source relative to the query's receiver pose.

Depth estimation. Following VisualEchoes [50], we use a U-net [106] that takes the log magnitude spectrogram of an echo as input and predicts the depth map (Sec. 5.4 in main) as seen from the echo's pose. The encoder of our U-net has 6 layers. The first layer is a convolution with a kernel size of 3, stride of 2, padding of 1 along the height dimension, and 2 input channels. The remaining 5 layers are convolutions with a kernel size of 4, padding of 1 and stride of 2. These 5 layers have 64, 64, 128, 256 and 512 input channels, respectively. Each convolution is followed by a ReLU activation [102, 103] and a BatchNorm [105].

The decoder of the U-net has 5 transpose convolution layers. Each transpose convolution has a kernel size of 4, stride of 2 and input padding of 1. Except for the last layer that uses a sigmoid activation function to generate depth maps, which are normalized such that all pixels are in the range of $[0, 1]$, each transpose convolution has a ReLU activation [102, 103] and a BatchNorm [105]. The decoder layers have 512, 1024, 512, 256 and 128 channels, respectively. We use skip connections between the encoder and the decoder starting with their second layer. We switch off bias in both the encoder and decoder.

A.4 Model architectures for our audio-visual policy

Global map encoder. Our global map encoder takes in the current global occupancy map $\mathcal{M}_t^\pi$ (Sec. 4.2.1 in main) as input, and feeds it to an ResNet-18 [86] model, which outputs a 512-dimensional feature. We train the ResNet from scratch.

Visual Encoder. Our visual encoder takes in the current RGB-D image V_t^π (Sec. 4.2.1 in main) as input, and first applies as 2D Conv layer with 3 output channels. We then feed this intermediate output to an ResNet-18 [86] model, which outputs a 512-dimensional feature. We train the ResNet from scratch.

Pose Encoder. We use the same architecture as in our neural acoustic rendering model (Sec. A.3.1) for encoding the agent pose.

Audio Encoder. Our audio encoder takes in the binaural echo response's log magnitude spectrogram A_t^π (Sec. 4.2.1 in main) as input, and feeds it to an ResNet-18 [86] model, which outputs a 512-dimensional feature. We train the ResNet from scratch.

We concatenate all the features from above and pass them through a stack of linear layers to obtain the current step's aggregated feature. This feature is 512-dimensional.

Policy network. Our policy network (Sec. 4.2.2 in main) comprises a GRU and an actor and a critic network. We use a single bidirectional GRU with a 512-dimensional hidden feature and a single recurrent layer. At each step, the GRU takes the corresponding aggregated feature as input.