A.1 In-the-wild Evaluation

Video results. We provide some random samples of prediction results from our in-the-wild dataset on our webpage. We also provide qualitative results for motion correlation along with our predicted time difference on our webpage. Please click here for videos. When watching, we recommend wearing headphones, since it can be difficult to perceive stereo sound without them.

A.2 Visually-guided Time delay Estimation

Video results. We provide more qualitative results for the visually-guided speaker localization task along with audio on our webpage. When watching, we recommend wearing headphones, since it can be difficult to perceive stereo sound without them.

A.3 Qualitative Results on Phone Recordings

We also ran our model on ordinary iPhone-recorded videos, exploiting the fact that portrait-mode recordings have a sufficiently large baseline for estimating time delays. We use a combination of self-collected videos and internet videos (we used an iPhone 12 for self-recorded videos and searched Flickr for videos with tags indicating that they were recorded with an iPhone 13). We provide qualitative results in Fig. 9. Please see our webpage for more video results.

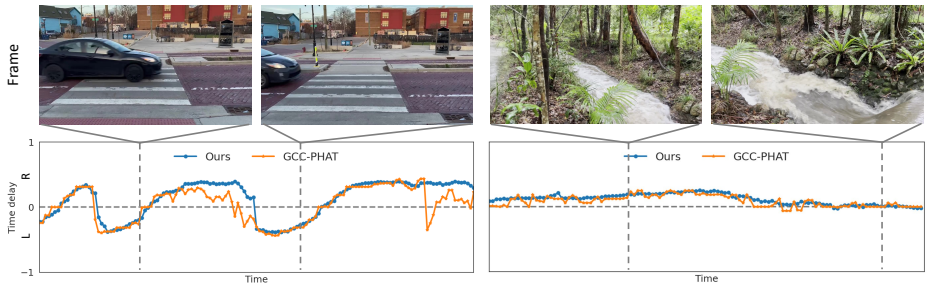
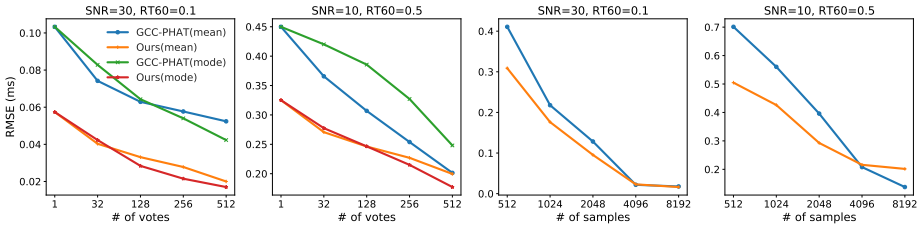


Fig. 9: **Qualitative results for iPhone videos.** We show time delays both for our method and for GCC-PHAT over time. The left video shot by the authors records several vehicles driving from left to right. The right video, from Flickr user *Black Diamond Images*, shows a waterfall that is recorded by a moving camera. In the first frame, the waterfall is to the right. The camera then moves to face it directly.

A.4 Ablation Study

Post-processing. We study the effect of the number of votes, m, used during post-processing for both StereoCRW and GCC-PHAT. We evaluate 1024-sample audio using $m \in \{1, 32, 128, 256, 512\}$ for both *mean* and *mode*. In the special case $m = 1$, the result is not affected by post-processing, and purely measures the quality of the representation. As shown in Fig. 10(a), both methods improve with the number of votes. Our method benefits from *mode* post-processing, while



(a) Post-Process (b) Input duration

Fig. 10: **Ablation experiments on the simulated data.** (a) We evaluate with different vote numbers and post-processing methods. (b) We evaluate longer audio lengths. We note that the x-axis of both plots is on a log scale.

mean works better for GCC-PHAT. In particular, we significantly outperform GCC-PHAT with $m = 1$ vote, emphasizing the quality of our representation.

We also design experiments to study the performance gap between using probabilistic post-processing and nearest neighbor (argmax) for our method. For consistency, we evaluated our StereoCRW model with both types of post-processing on two simulated environments, with different vote numbers. The results are shown in Fig. 11. Probabilistic post-processing outperforms the nearest neighbor post-processing in the complex simulated environment. We use probabilistic post-processing in our other experiments.

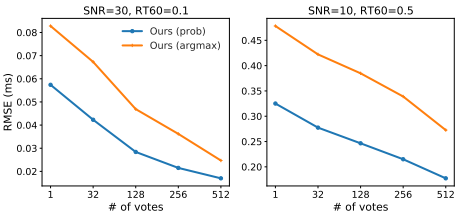


Fig. 11: Probabilistic vs. Argmax.

Duration. We ask how our model performs when given longer audio, exploiting the fact that our embeddings use fully convolutional networks and thus can be tested on arbitrary-sized inputs (Fig. 10(b)). We provided our method with various input sizes (up to 8× the training duration). For very long audio (4× training duration), we found that our model's performance starts saturating, and that GCC-PHAT overtakes it. This may be due to the fact that the model has a fixed-size ($d = 128$) representation, while GCC-PHAT grows its representation—the waveform itself—with the input size and eventually converges to the correct solution (a maximum likelihood estimate [51,87] in many situations).

Simulated vs. real data. We also study the data distribution gap between simulation and real-world data. We trained our best self-supervised model (StereoCRW) and Salvati et al. [1] on the simulated data with Free-Music-Archive clips. We evaluated on both simulated and in-the-wild recordings (Tab. 5). As expected, our model trained on FMA-Sim obtains competitive performance, but overall does not perform as well as a model trained on real data. The supervised model improves on the in-the-wild evaluation while the performance drops on the simulated evaluation cases when training on FMA-Sim. We also include the comparison between mode and mean post-process for Salvati et al. [1] in the Tab. 5.

Table 5: Time delay estimation on simulated data and in-the-wild recordings. *Vox-Sim* is the simulator [61] with VoxCeleb2 clips and *FMA-Sim* is the simulator with Free-Music-Archive clips.

					Simulation		Real-world
Model	Variation	Data	Num	Aug	MAE	RMSE	Acc (%)
Salvati et al. [1]	Mode	Vox-Sim	8K		0.146	0.306	87.6
	Mean	Vox-Sim	8K		**0.126**	**0.254**	87.0
	Mode	Vox-Sim	8K	✓	0.184	0.327	87.5
	Mean	Vox-Sim	8K	✓	0.169	0.294	87.3
	Mode	FMA-Sim	95K		0.150	0.294	88.5
	Mean	FMA-Sim	95K		0.135	0.256	87.9
	Mode	FMA-Sim	95K	✓	0.160	0.303	**89.3**
	Mean	FMA-Sim	95K	✓	0.146	0.267	89.1
StereoCRW	Mode	Vox-Sim	8K	✓	0.193	0.360	85.6
StereoCRW	Mode	FMA-Sim	95K	✓	0.194	0.341	87.9
StereoCRW	Mode	FMA	95K	✓	**0.133**	**0.259**	**88.7**

A.5 Training with Youtube-ASMR

We also train our models on **YouTube-ASMR**, a highly diverse dataset of 30K binaural (83 hours) internet videos [84]. As the results shown in Tab. 6, our proposed ZeroNCE and StereoCRW methods outperform GCC-PHAT. Augmentation was important for YouTube-ASMR particularly, which failed to learn a good representation without it, perhaps due to the high diversity of the dataset.

Table 6: **Delay estimation on simulated data.** We use SNR = 10 and RT_{60} = 0.5s. ASMR is YouTube-ASMR [84]. Errors in ms. *Sup* refers to supervision.

Model	Variation	Data	Sup	Aug	MAE	RMSE
Salvati et al. [1]	Mean	Vox-Sim	✓		**0.126**	**0.254**
	Mean	Vox-Sim	✓	✓	0.169	0.294
GCC-PHAT [51]	Mean	–			**0.160**	0.318
Ours	Random	–			0.448	0.505
	MonoCLR	ASMR			0.425	0.620
	MonoCLR	ASMR		✓	0.177	0.330
	ZeroNCE	ASMR			0.349	0.468
	ZeroNCE	ASMR		✓	0.184	**0.313**
	StereoCRW	ASMR			0.736	0.913
	StereoCRW	ASMR		✓	**0.162**	0.315

A.6 Simulation Setup

We provide the details of each simulated room with its dimension and microphone positions in Tab. 7.

Table 7: **Simulation setup.** The unit is in meters.

	Room 1	Room 2	Room 3
Room dim (X, Y, H)	(7, 6, 3)	(4, 7, 2.8)	(7, 7, 2.7)
Left Mic position (X, Y, H)	(3.4, 1, 1.6)	(0.2, 3.2, 1.7)	(3.4, 3.1, 1.5)
Right Mic position (X, Y, H)	(3.7, 1, 1.6)	(0.2, 3.0, 1.7)	(3.5, 2.9, 1.5)
Source angle range	$[-90°, 90°]$	$[-90°, 90°]$	$[-90°, 90°]$
Source distance range	$[0.5, 3.0]$	$[0.5, 3.0]$	$[0.5, 3.0]$

A.7 Implementation Details

Network architecture. We use a ResNet [38] with 9 layers as the backbone for the audio encoder. We modify the input channel number of the first convolution layer to be 2, and the output of the last fully-connected layer as 128. For a raw waveform of length L, we use an STFT with a window length of 256 and hop length of $\lfloor \frac{L}{128} \rfloor$ to create an input spectrogram. For the audio-visual task, we use a hop length of 160 to create an input spectrogram.

Augmentations. During the training, we apply the following augmentations to audio where the first three are regular augmentations applied to all the models and the last two are applied to augmented models only:

- Random channel swapping: we randomly swap the left and right audio channels with a probability of 0.5.
- Random channel-wise scaling: we randomly re-scale each audio channel by the factor in the range of $[0.5, 1.5]$.
- Random shifting: for the instance discrimination model with mono audio, we randomly shift the audio for -16 to 16 samples. For the audio-visual model, we apply different random shifts for each mono audio with -24 to 24 samples.
- Random noise: we add random Gaussian noise to audio with SNR=$[0, 30]$.
- Random reverberation: we add random reverberation to audio with RT_{60}=$[0, 0.9]$.
- Mixture augmentation: we randomly add another sound to the original audio. We normalize the second sound to be 10% – 100% loudness of the original audio before mixing. For the audio-visual model, the second sound is normalized to be 50%-150% intensity level of the original one.

When computing affinity matrix A_{21} for the contrastive random walk model, we do not augment $\mathbf{x}_1$ with noise or mixture augmentation, so as to avoid learning unexpected matching. Similarly, for the instance discrimination model, we do not apply noise, reverberate or mixture augmentations to one of the two channels.

Training details. To accelerate the training process, we first train each model with 0.48s audio (7680 samples) and then finetune on the input audio of 1024 samples using a correspondingly finer hop length.