Supplementary Material for: Structure from Silence

A Dataset Examples

We show video samples of *static* recordings (`static_near.mp4` and `static_far.mp4`) and video samples of *motion* recordings (`motion.mp4`) along with audio.

B Additional robot navigation experiments

We perform additional robotic navigation experiments. For these, we use a robot equipped with to avoid the need for the robot to rotate to sense both sides of the scene (the model is otherwise unchanged).

Experiment setting. We evaluate our model in 3 additional rooms. Their floor plans and track designs are shown in Fig. 11. Similarly, the robot starts from one of several unknown locations (40cm, 80cm, or 120cm from the wall) and orientations (30° left or right, or facing forward). For each room, we repeat the experiment 9 times and measure the longest distance along the track it attained before colliding with a boundary. For the 'L'-shaped track in Room #2, we determine the distance by projecting the final position to the center line of the track. All rooms (4 in total) are located in distinct buildings, and are not included in the training set.

Algorithm. The detailed algorithm to control the robot is shown in Algorithm 1. When computing the near-wall probability $P(L)$ and $P(R)$, we average the predictions of 20 clips of 3s audio to make the prediction more accurate robust. For each room, we manually adjusted the threshold p, which accounts for per-room biases in the classifier's output. **Experimental results.** As before, we evaluate using the "straight line" policy. We measure the performance by computing the percentage of the completed track, and take the average of all trials, as well as the average of the trials starting with the same orientation, as shown in Fig. 10. It can be seen that the "straight line" policy performs well when reaching the goal does not require turning, as expected. However, it performs significantly worse than our model when the robot starts at an angle that does not follow the wall. In contrast, the distance our model traverses is more consistent between starting orientations, indicating that the model more successfully guides the robot to navigate along the wall. When looking at the average performance with different starting orientation on individual rooms, we also find that the "straight line" policy only outperforms our method when starting facing forward on straight tracks, and performs worse than ours with any other starting orientation or on the 'L'-shaped track in Room #2.

Vision model. While, in principle, vision-based navigation is very effective (and is capable of significantly outperforming audio-based navigation), we found in initial experiments that our vision-based model did not generalize well to novel viewpoints, perhaps because the RGB-D training only contained wall-facing viewpoints. To address this, we collected 12K RGB-D images as training data from the same scenes, with additional random viewing directions, and fine-tune our visual model on it. We then used a slightly modified algorithm (Algorithm 2). To keep the control algorithm as close as possible to the controller for audio model, we made a small modification to Algorithm 1. We estimated the turning angle based on the prediction of near-wall side (which can be obtained by comparing left-side score and right-side score at the initial step). The reason for this is to avoid the noise from the prediction of far-wall side, considering the threshold p is usually very high (we set it as 0.97 in the experiment).

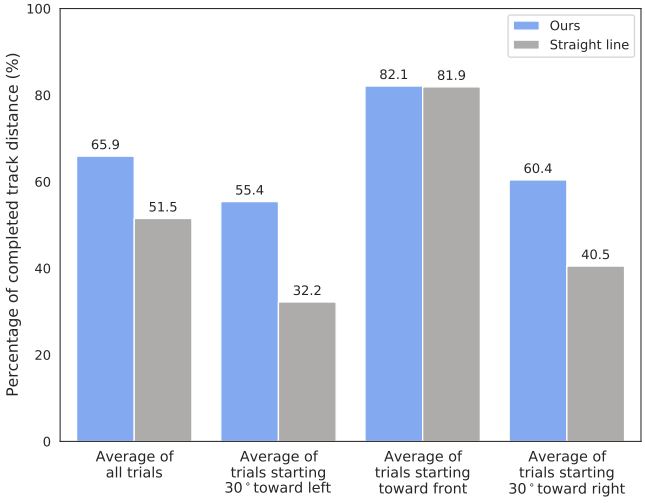
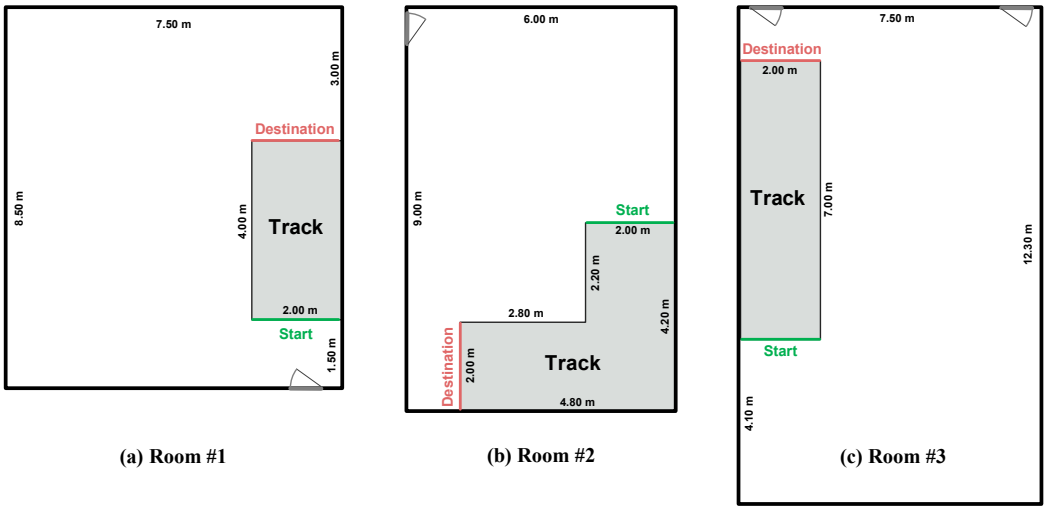


Figure 10: Additional Robot navigation results.

Video demo. We provided demo video (`robot_demo.mp4`) in the supplementary material. Please check the video for qualitative results of robot navigation.



(a) Room #1 (b) Room #2 (c) Room #3

Figure 11: Classroom floor plans and corresponding track settings.

C Obstacle detection on static-dense recordings

We evaluated the obstacle detection task for *static-dense* recordings by training a model on 3 rooms and testing on an unseen room with the same method as the main paper. To evaluate whether the success rate for wall perception increases when participants have self-motion, we simulate the motion of the recorder by concatenating two audio with the same angle from adjacent grids, and predict whether the agent is moving towards or far away from the obstacles. We report the performance of the model (**Ours-static**) trained on *static* recordings for comparison as well.

Table 6: **Obstacle detection** for *static-dense* recordings. Here, **S** denotes *static* and **M** means *motion*.

Model	Task	AP(%)	Acc(%)
Chance	S	46.4	50.0
Chance	M	52.4	49.8
Ours	S	**96.4**	**83.9**
Ours	M	**98.3**	**94.2**
Ours-static	S	93.9	84.6

As shown in Table 6, the average precision above 90% in *static-dense* recordings with both settings. Comparing experimental results of with or without motion, we can find the network predicts more accurately with the simulated motion. We also evaluate our *static-dense* model on *static* recordings, finding that it could obtain 62.8% AP and 60.9% accuracy (on par with the equivalent model trained on the *static* dataset). A video demo is available in the supplementary material (`obstacle_detect_video_demo.mp4`).

D Indoor localization on static-dense recordings

We also asked whether ambient sounds can convey the absolute position in rooms (rather than distance to walls), when trained and tested on similar (or the same) room. We performed a localization experiment for *static-dense* recordings. We use the same network as the obstacle detection task and replace the last linear layer with a N-class fully-connected layer, where N is the number of grid cells in the room. We formulate this as a multi-way classification task and predict which grid the given audio comes from. We train the model with the cross entropy loss and evaluate it with top-1, top-5, and average distance (*i.e.*, the Euclidean distance between our predicted and target grid position).

Grid classification. We ask to what extent ambient sound can allow a model to predict which grid cell within the *same room* the model was trained on. For the train/test split, we randomly select audio samples with three angles in each grid cell as training, then test on sound with the remaining angle (i.e. the test samples are unseen, but the model has been trained with other examples from the same room). The results are shown in Table 7. The model's accuracy is significantly better than chance. As expected, we see that our absolute-position model does not generalize well to other rooms, which may be due to both ambiguities in the coordinate system and fundamental ambiguities in the prediction problem.

Generalization over time. Next, we evaluated our model's ability to predict the absolute position in the room using the audio samples recorded at different times to see how well the model could generalize over time. In this experiment, training and test sets are from the same room but different times. The time intervals for Room #4, #5 and #6 are 1 week, 1 week, and 1 hour, respectively.

We show the experimental results in Table 7. It can be seen that the performance is lower when audio is recorded at different times, confirming the assumption that the network can indeed use the shortcut

Table 7: **Indoor Localization** for *static-dense* recordings.

EXP.	Random	Room 4 Non time-shift	Time-shift	Random	Room 5 Non time-shift	Time-shift	Random	Room 6 Non time-shift	Time-shift
Top-1 (%)	4.00	32.3	18.3	2.86	29.5	18.8	4.00	34.3	22.8
Top-5 (%)	20.0	77.7	50.3	14.5	70.8	51.8	20.0	74.3	55.6
Avg. Distance	3.17	1.23	1.71	3.59	1.98	2.17	3.17	1.31	1.87

of memorizing transient background sounds. However, the performance is still significantly better than random chance for recordings that were taken a week apart.

E Relative depth order on motion recordings

We design experiments to investigate the accuracy of relative depth estimation model as the magnitude of the depth difference changes. We paired audio in the test set with 5 different distance ranges, and re-evaluated our model (without re-training). We used the variations of our networks that are initialized randomly. The results are shown in Figure 12. We see that prediction accuracy increases as the relative depth increases, and that the models are more confident when the distances apart are at least 0.2 meters.

F Generalization between *motion* and *static* recordings

We give the results of the model trained on motion recordings (recorded in hallways) and tested on static recording (recorded in the classrooms). The experiment results are shown in Table 8 . Our model can still perform well above random chance despite evaluating on a different domain.

Table 8: **Generalization between *motion* and *static* recordings** on obstacle detection and depth order task.

	AP(%)	Acc(%)
Obstacle detection	61.3	51.9
Depth order	78.8	70.2

G Effects of Non-ambient sounds

To investigate how non-ambient sounds from environment could affect our prediction, we mix static recordings with sounds from FreeSound [81] to simulate other sound sources in the scenes. By using synthetic mixtures, we prevent our model from exploiting other cues, *e.g.* echolocation. We use 200 types of sound events at training and test time, *e.g.*, human sounds, animal, and music. We test our models with (or without) re-training on the mixed sounds. The results are shown in Fig. 13. We can see that the performance drops as the volume of distracting sound increases, while still outperforming chance. Re-training the model with mixed sounds improves performance and makes the model significantly more robust.

H Conditional absolute depth estimation

Implementation. We use a Siamese network from relative tasks to build conditional absolute depth (**CAD**) model. The model takes three inputs: audio s_1, reference audio s_2, and ground-truth depth x_2. We first extract audio features with VGGish backbone and calculate the depth embedding following [82]. We add the depth embedding and reference audio feature and fuse features via concatenation prior to the multi-layer perceptron (*i.e.*, FC-ReLU-FC-ReLU-FC layers). During training, the reference sounds are randomly selected from the same scene, while those are fixed for each sample in the test set (Fig. 14).

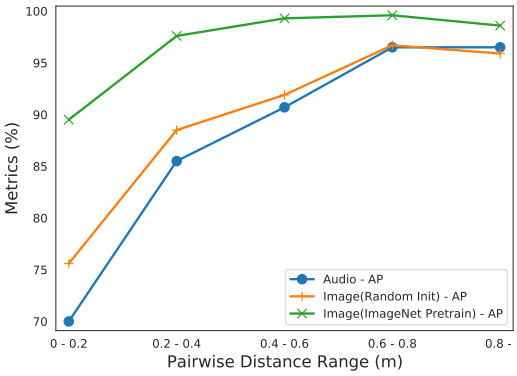


Figure 12: Relative depth order performance vs. distance.

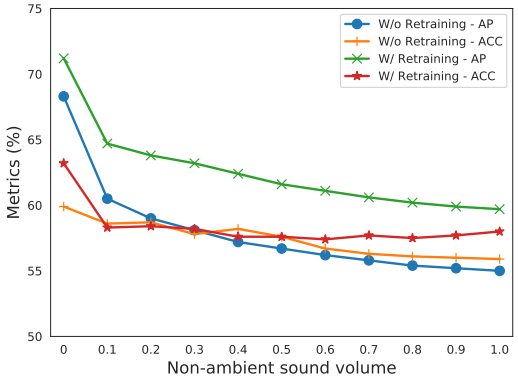


Figure 13: Obstacle detection performance vs.volume of distracting sound.

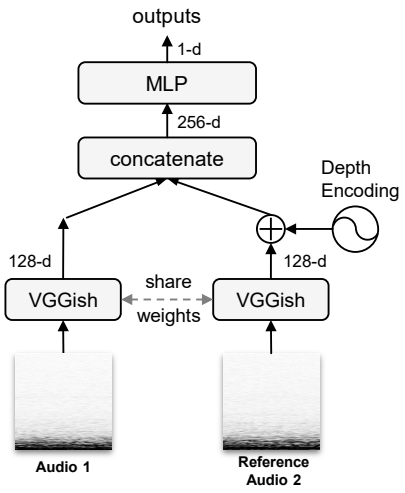


Figure 14: Conditional absolute depth model.

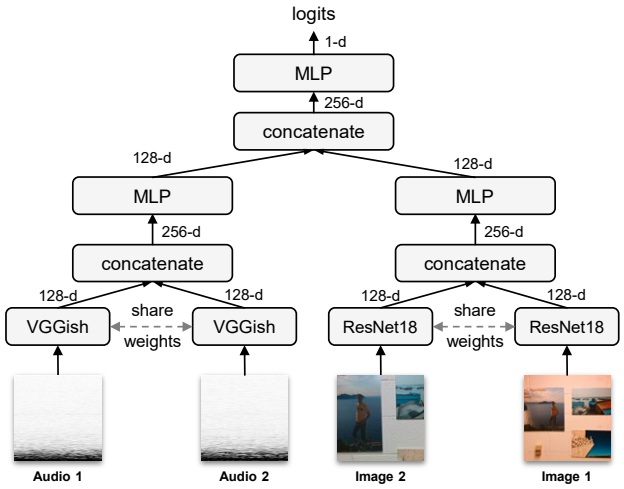


Figure 15: The detail of AV-Order model.

I Self-supervised audio-visual learning

Implementation. We approach the AV-Order model by building two Siamese networks for both audio and visual modalities. As shown in Fig. 15, for each modality, we first extract features from inputs and concatenate them prior to the multi-layer perceptron. Then we concatenate two 128-d feature vectors from image and audio branches, and forward it to an MLP (*i.e.*, FC-ReLU-FC layers, and all the MLP blocks are the same) for the logits.

AudioSet features. The low performance of the AudioSet features in Table 4, 5 could be due to differences in the types of sounds considered, as well as the preprocessing in each method. The preprocessing procedure used in the AudioSet model (which is standard on the dataset) throws away all but the 125Hz-7500Hz frequency range, which our analysis (Section 5.2) suggests is important. To test this, we trained a random feature baseline with (and without) this preprocessing, and see a large difference in their performances (Table 4, 5).

Few-shot learning. To help understand how audio-visual self-supervision improves fine-tuned models, we measure the performance using various numbers of labeled examples as training set (5%, 10%, 20%, 40%, 100%). The experiment results are shown in Figure 16. We can see a large improvement from self-supervised initialization when there is more unlabeled data than labeled data, especially in few-shot training regimes. When the number of labeled examples approaches the number of unlabeled examples, the trained-from-scratch model catches up. This is expected, since the two datasets are exactly the same, and the labels provide strictly more information than the audio.

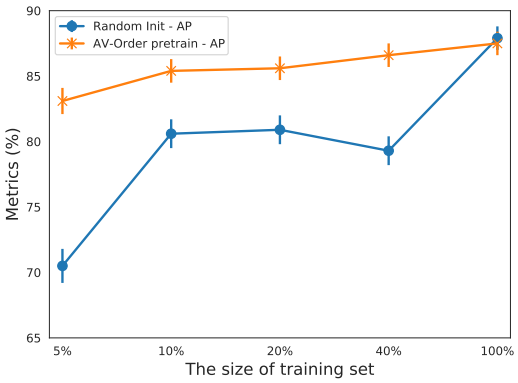


Figure 16: Relative depth order performance vs. training set size.

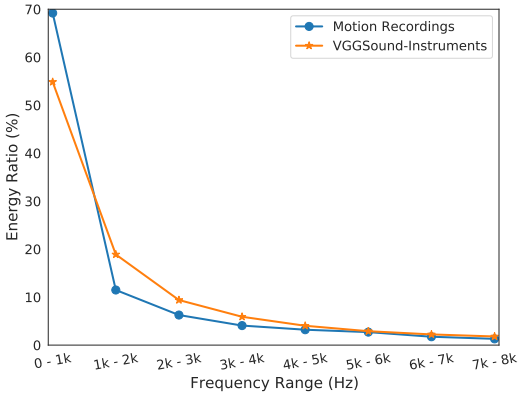


Figure 17: Energy ratio vs. different frequency.

J VGGSound-Instrument dataset

We sampled 37 classes of musical instruments with 32k video clips of 10s length from VGGSound dataset [80], and we call this subset VGGSound-Instruments. Video classes are listed as below. We explore frequency energy distributions of the motion recordings and VGGSound-Instruments by computing the energy ratio, *i.e*. the ratio between the magnitude in a given frequency and the sum of all frequency magnitudes, from all the samples. As shown in Fig. 17, we can see that 70% of the spectrogram energy is concentrated on low-frequencies (0 to 1000Hz), which may explain why our model achieves the best performance with low-frequency information in Fig. 8. Furthermore, our ambient dataset contains relatively more low-frequency sounds than VGGSound-Instrument, suggesting the diversity of our dataset.

playing accordion	playing acoustic guitar	playing banjo	playing bass drum	playing bass guitar
playing bongo	playing cello	playing clarinet	playing congas	playing cornet
playing cymbal	playing djembe	playing double bass	playing drum kit	playing electric guitar
playing electronic organ	playing erhu	playing flute	playing glockenspiel	playing guiro
playing hammond organ	playing harp	playing harpsichord	playing mandolin	playing marimba, xylophone
playing piano	playing saxophone	playing sitar	playing snare drum	playing steel guitar, slide guitar
playing tabla	playing timbales	playing trumpet	playing ukulele	playing vibraphone
playing violin, fiddle	playing zither			

Algorithm 1 Policy for navigating along the wall using ambient sound

Require: threshold p, α; step length v
while No crash **do**
  Input: left and right side sound
  $P(L)$ = Network(left sound)
  $P(R)$ = Network(right sound)
  Angle $\propto |0.5(P(L) + P(R)) - p|$
  if $|P(L) - P(R)| < \alpha$ **then**
    Go forward
  else
    if $0.5(P(L) + P(R)) > p$ **then**
      # Robot might be near to the wall
      if P(L) > P(R) **then**
        Turn right and go forward
      else
        Turn left and go forward
      end if
    else
      # Robot might be far from the wall
      if P(L) > P(R) **then**
        Turn left and go forward
      else
        Turn right and go forward
      end if
    end if
  end if
end while

Algorithm 2 Policy for navigating along the wall using RGB images

Require: threshold p; step length v
Determine which side is near-wall side
Input: left and right RGB image
$P(L)$ = Network(left RGB image)
$P(R)$ = Network(right RGB image)
if $P(L) < P(R)$ **then**
  Near-wall side is on the robot's right; Far-wall side is on the robot's left
else
  Near-wall side is on the robot's left; Far-wall side is is on the robot's right
end if
while No crash **do**
  Input: left and right RGB image
  $P(N)$ = Network(near-wall side RGB image)
  Angle $\propto |P(n) - p|$
  if $P(N) > p$ **then**
    # Robot might be near to the wall
    Turn to the far-wall side and go forward
  else
    # Robot might be far from the wall
    Turn to the near-wall side and go forward
  end if
end while