Chat2Map: Efficient Scene Mapping from Multi-Ego Conversations Supplementary Material

Sagnik Majumder[1,2,3] Hao Jiang[2] Pierre Moulon[2] Ethan Henderson[2]
Paul Calamia[2] Kristen Grauman[1,3*] Vamsi Krishna Ithapu[2*]
[1]UT Austin [2]Reality Labs Research, Meta [3]FAIR

In this supplementary material we provide additional details about:

- Video (with audio) for qualitative illustration of our task and qualitative assessment of our map predictions (Sec. 1)
- Societal impact of our work (Sec. 2), as mentioned in Sec. 6 in main
- Detailed analysis of the power and time cost of our model (Sec. 3), as mentioned in Sec. 5.1 in main
- Experiment to show the effect of ambient environment sounds on mapping accuracy (Sec. 4), as referenced in Sec. 5.2 in main
- Experiment to show the effect of *unheard* sounds on map predictions (Sec. 5), as noted in Sec. 5.2 in main
- Experiment to show the impact of the visual budget B (Sec. 3 in main) on mapping quality (Sec. 6), as referenced in Sec. 5 and 5.2 in main.
- Experiment to show the effect of sensor noise on mapping accuracy (Sec. 7), as mentioned in Sec. 5 and 5.2 in main.
- Experiment to show mapping performance as a function of the target map size (Sec. 8), as noted in Sec. 5.2 in main.
- Experiment to show the effect of different types of ego initializations on map predictions (Sec. 8), as referenced in Sec. 5.2 in main
- Experiment to show the effect of multiple random dataset splits on mapping quality (Sec. 8), as referenced in Sec. 5.2 in main
- Experiment to show the effect of the training data size on the model performance (Sec. 8), as mentioned in Sec. 5 in main
- Dataset details (Sec. 12) in addition to what's provided in Sec. 5 in main.
- Additional baseline details for reproducibility (Sec. 13), as referenced in Sec. 5 in main.
- Architecture and training details (Sec. 14), as noted in Sec. 5 in main.

*Equal contribution

1. Supplementary video

The supplementary video qualitatively depicts our task, Chat2Map:Efficient Scene Mapping from Multi-Ego Conversations. Moreover, we qualitatively show our model's mapping quality by comparing the predictions against the ground truths and the visual samples chosen by our sampling policy for efficient mapping, and analyze common failure modes of our model. We also demonstrate the acoustically realistic SoundSpaces [1] audio simulation platform that we use for our core experiments. Please use headphones to hear the spatial audio correctly. The video is available on http://vision.cs.utexas.edu/projects/chat2map.

2. Societal impact

Our model enables efficiently mapping a scene from natural conversations. This has multiple applications with a positive impact. For example, accurate mapping of an unseen environment enables many downstream applications in AR/VR (e.g., accurate modeling of scene acoustics for an immersive user experience) and robotics (e.g., a robot using a scene map to better navigate and interact with its environment). However, our model relies on speech inputs, which when stored or used without sufficient caution could be prone to misuse by unscrupulous actors. Besides, the dataset used in our experiments contains indoor spaces that are predominantly of the Western design, and with a certain object distribution that is common to such spaces. This may bias models trained on such data toward similar types of scenes and reduce generalization to scenes from other cultures. More innovations in the model design to handle strong

Model	F1 score ↑	IoU ↑
All-occupied	63.4	48.8
Register-inputs	72.6	60.1
OccAnt [13]	74.5	62.7
AV-Floorplan [12]	78.7	67.5
Ours	**81.9**	**71.5**
Ours w/o vision	73.5	61.2
Ours w/o audio	78.1	66.7
Ours w/o $E_i^{'}$'s speech	81.5	70.9
Ours w/o shared mapping	80.0	69.1

Table 1. *Passive mapping* performance (%) with ambient sounds.

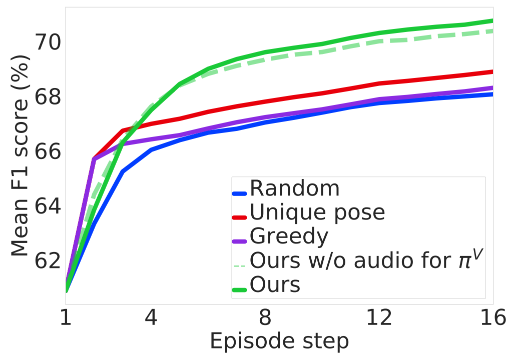


Figure 1. Effect of ambient environment sounds on *active mapping*

shifts in scene layout and object distribtutions, as well as more diverse datasets are needed to mitigate the impact of such possible biases.

3. Power and time cost

With visual budget $B = 2$ and episode length $T = 16$ (Sec. 3 and 5 in main), our model skips 28 frames and saves 7.2 GFLOPs in mapping but adds 24.1 GFLOPs due to the policy, thus adding a net of 16.1 GFLOPs. This translates to around 0.5 Watt [4] of extra power for running the active mapper but a saving [9] of $\sim 28 \times 3 = 74$ Watt in camera capture, thus saving a net of ~73.5 Watt in power. The total runtime of our model in this setting is 5.6 s on a Quadro GV100 GPU.

Compared to the heuristical baseline policies (Sec. 5 in main), our policy adds ~24GFLOPs, consuming 1 extra Watt [4] on modern GPUs, while improving the map by 14 m^2 on avg (Fig. 3 in main).

4. Ambient and background sounds

We also test our model's robustness to ambient and background sounds by inserting a non-speech sound (*e.g*. running AC, dog barking, etc.) at a random location outside the egos' trajectories. Although quite challenging, our model performs better than the baselines for both *passive* (Table 1) and *active mapping* (Fig. 1). Hence, even without explicit audio separation, our model is able to implicitly ground its audio representations in the corresponding pose features for accurate mapping.

Model	F1 score ↑	IoU ↑
All-occupied	63.4	48.8
Register-inputs	72.6	60.1
OccAnt [13]	74.5	62.7
AV-Floorplan [12]	79.0	67.7
Ours	**81.6**	**71.1**
Ours w/o vision	72.6	60.1
Ours w/o audio	78.1	66.7
Ours w/o $E_i^{'}$'s speech	81.3	70.7
Ours w/o shared mapping	80.7	70.0

Table 2. *Passive mapping* performance (%) on *unheard* sounds.

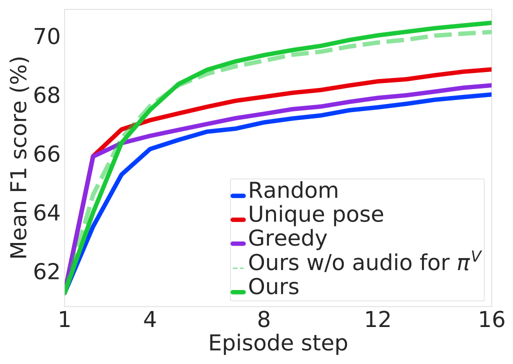


Figure 2. *Active mapping* performance vs. episode step on *unheard* sounds.

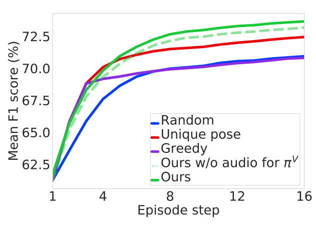
5. Unheard sounds

In Sec. 5.1 in main, we showed results with *heard* sounds (Sec. 5 in main), *i.e*. the *anechoic* speech sounds uttered by the egos are shared between train and test splits. However, due to our use of *unseen* environments in test (Sec. 5 in main), the spatial speech sounds input to our model during test are not heard in training. To make the evaluation even more challenging, we conduct a parallel experiment here, where even the anechoic speech is distinct from what's used in training, which we call as the *unheard* sound setting (Sec. 5 in main).

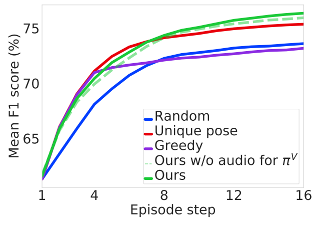
Table 2 shows our *passive mapping* results in the *unheard* sound setting. Our model is able to retain its performance margins over all baselines even in this more challenging scenario.

We notice a similar trend upon evaluating our model for *active mapping* on *unheard* sounds. Fig. 2 shows that our model is able to generalize to novel sounds better than all baselines.

This indicates that both our mapper f^M and visual sampling policy π^V are able to learn useful spatial cues from



(a) $B = 4$



(b) $B = 6$

Figure 3. *Active mapping* performance vs. episode step with $B \in \{4, 6\}$.

Model	F1 score ↑	IoU ↑
All-occupied	63.0	48.3
Register-inputs	72.3	59.7
OccAnt [13]	74.7	63.0
AV-Floorplan [12]	77.6	65.8
Ours	**79.1**	**68.0**
Ours w/o vision	72.6	60.0
Ours w/o audio	76.7	65.1
Ours w/o $E_i^{'}$'s speech	78.8	67.7
Ours w/o shared mapping	78.5	67.2

Table 3. *Passive mapping* performance (%) with sensor noise.

audio that are agnostic of the speech content and semantics.

6. Visual budget value

So far, we have shown *active mapping* results with the visual budget set to $B = 2$ (Sec. 5.1 and Fig. 3 in main). To analyze the effect of larger values of B, we show our *active mapping* performance for $B \in \{4, 6\}$ in Fig. 3. Our model outperforms all baselines even for these larger B values. We also observe that the lower the visual budget, the higher the performance margins are for our model. This shows that our model is particularly more robust to the lack of visuals in extremely low-resource settings.

7. Sensor noise

Here, we test our model's robustness to sensor noise by adding noise of the appropriate type individually to each sensor. For RGB images, we sample the noise from a Gaussian distribution with a mean of 0 and a standard deviation of 0.2 [13, 14]. For depth, we use the Redwood depth noise model [2, 13, 14], where the amount of noise is higher for higher depth values and vice-versa. Following [13], we sample pose noise from a truncated Gaussian with a mean of 0.025 m and a standard deviation of 0.001 m for the spatial location component of an ego pose $((x, y)$ in Sec. 3 in main$)$. For orientation θ (Sec. 3 in main), we use another truncated Gaussian with a mean of 0.9° and a standard deviation of 0.057°. Both distributions are truncated at 2 standard deviations. For our multi-channel microphones (Sec. 3 in main), we add a high amount of noise (SNR of 40 dB) [1] using a standard noise model [2, 19].

Table 3 and Fig. 4 report our *passive* and *active mapping* performance, respectively, in the face of sensor noise. In both settings, although our model's performance declines in comparison to the noise-free setting (cf. Table 1 and Fig. 3 in main), it generalizes better than all baselines, thereby underlining the effectiveness of our method.

8. Target map size

In main (Sec. 5.1), we showed mapping results with $H \times W = 6.4 \times 6.4$ m^2($\sim$ 41 m^2), where H and W denote

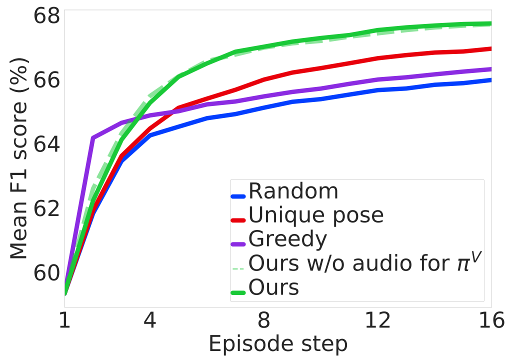
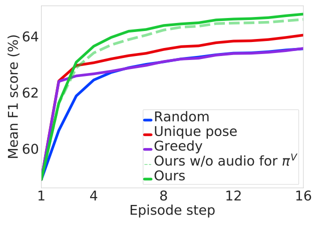


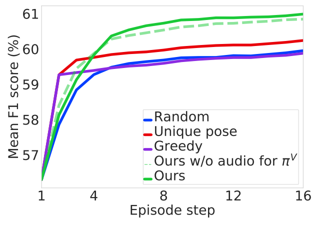
Figure 4. *Active mapping* performance vs. episode step with sensor noise.

Model	$H = W = 8$ m F1 score ↑	$H = W = 8$ m IoU ↑	$H = W = 9.6$ m F1 score ↑	$H = W = 9.6$ m IoU ↑
All-occupied	53.5	37.9	46.4	31.2
Register-inputs	65.9	53.4	61.6	49.6
OccAnt [13]	67.8	55.7	63.0	51.3
AV-Floorplan [12]	71.4	59.1	68.7	53.1
Ours	**73.4**	**60.7**	**72.0**	54.4
Ours w/o vision	66.1	53.5	62.6	50.3
Ours w/o audio	71.1	58.1	63.8	51.3
Ours w/o $E_i^{'}$'s speech	73.3	60.5	67.6	54.0
Ours w/o shared mapping	72.9	60.3	68.0	**54.5**

Table 4. *Passive mapping* performance (%) for larger target map sizes.



(a) $H = W = 8$ m

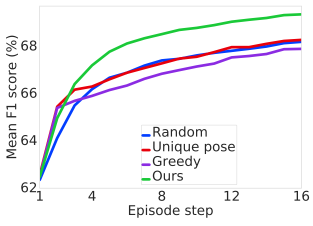


(b) $H = W = 9.6$ m

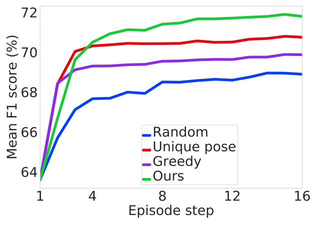
Figure 5. *Active mapping* performance vs. episode step for larger target map sizes.

Model	Facing away F1 score ↑	Facing away IoU ↑	Separated by occlusion F1 score ↑	Separated by occlusion IoU ↑
OccAnt [13]	75.2	63.7	75.3	62.8
AV-Floorplan [12]	79.6	68.7	80.1	69.0
Ours	**82.9**	**72.8**	**83.0**	**71.8**

Table 5. *Passive mapping* performance (%) for different ego initializations.



(a) Facing away



(b) Separated by occlusion

Figure 6. *Active mapping* performance for different ego initializations.

Model	F1 score ↑	IoU ↑
OccAnt [13]	74.4	63
AV-Floorplan [12]	78.8	67.6
Ours	**81**	**70.6**

Table 6. Average *passive mapping* performance (%) over 3 random data splits.

the height and width of the ground-truth local 360° FoV maps (Sec. 4.3 in main). To analyze the impact of larger target map sizes on the mapping quality, we also test our model with $H \times W \in \{8 \times 8 \text{ m}^2 (64 \text{ m}^2), 9.6 \times 9.6 \text{ m}^2 (\sim 92 \text{ m}^2)\}$. Table 4 and Fig. 5 show the corresponding results for *passive* and *active mapping*, respectively. In both cases, our model outperforms all baselines by a substantial margin, showing that our method is also robust to higher target map sizes.

9. Different ego initializations

Here, we study the effect of ego initialization by considering two cases: 1) the egos initially face away from each other, and 2) the egos are initially separated by an occlusion. In both cases, our model outperforms all baselines on both passive (Table 5) and active mapping (Fig. 6), showing that our model is robust to different ego initializations.

10. Multiple random dataset splits

Here, we gauge the effect of multiple random dataset splits on our model performance. In each random split, we use the same distribution of train/val/test scenes and episode counts as mentioned in Sec. 5 in main, but instantiate each episode with a new seed. Table 6 and Fig. 7 report the passive

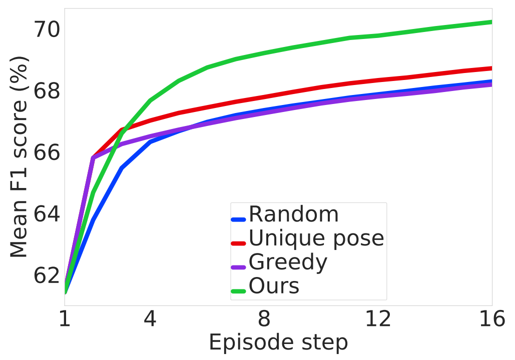


Figure 7. Average *active mapping* performance over 3 random data splits vs. episode step.

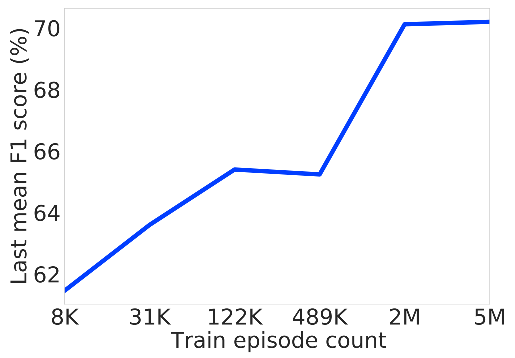


Figure 8. *Active mapping* performance on *unheard* sounds vs. training data size.

and active mapping performance averaged over 3 random splits, respectively. We observe that our model is more robust to different dataset instantiations than all baselines across both passive and active mapping.

11. Training data size

Here, we analyze the effect of the training data size on the active mapping quality. Fig. 8 shows the final active mapping F1 score as a function of the number of training episodes. The mapping performance significantly improves with more training data up to 2 million episodes and then flattens between 2 and 5 million episodes.

12. Dataset details

Here, we provide additional dataset details.

Visual data. All RGB-D images in our experiments have a resolution of 128×128.

To generate the topdown occupancy maps, we threshold the local pointcloud computed from the 90° FoV depth images (Sec. 4.1 in main) using a lower and upper height limit of 0.2 and 1.5 m, respectively, such that a map cell is considered occupied if there is a 3D point for it in the 0.2-1.5 m range, and free otherwise.

To generate an estimate of the scene map, we register the estimates of ground-truth local 360° FoV maps, $\tilde{M}_{i,j}$s onto a shared scene map $\tilde{M}$ (Sec. 4.3 in main) and maintain a count of the number of updates undergone by every cell in the shared map. To register a local estimate $\tilde{M}_{i,j}$, we first translate and rotate $\tilde{M}_{i,j}$ within $\tilde{M}$ on the basis of its normalized pose $P_{i,j}$. Next, we add $\tilde{M}_{i,j}$ with the corresponding part of $\tilde{M}$ and update the counter for every map cell that's been changed through the registration. We repeat this process for every $\tilde{M}_{i,j}$ in the episode. Finally, we normalize the updated $\tilde{M}$ by dividing each cell in it by its number of updates from the counter, and thresholding at 0.5. In our experiments, $\tilde{M}$ covers a maximum area of 128.4×128.4 m^2.

Audio data. For each conversation episode, we randomly choose 2 speakers from the same split – *heard* or *unheard* (Sec. 5 in main). Starting at a random time in the audio clip for each speaker, we choose contiguous 3s slices from each clip for T steps to use as the anechoic audio for the two egos in the episode, where T denotes the episode length (Sec. 3 in main). Further, we normalize each slice to have the same RMS value of 400 across the whole dataset, where all audio is sampled at 16 kHz and stored using the standard 16-bit integer format.

To generate the spectrograms, we convolve a speech slice with the appropriate 9-channel RIR sampled at 16 kHz and compute its STFT with a Hann window of 31.93 ms, hop length of 8.31 ms, and FFT size of 511 to generate 9-channel magnitude spectrograms, where each channel has 256 frequency bins and 257 overlapping temporal windows. We assume access to detected and separated speech from the egos at all times since on-device microphones of AR glasses can tackle nearby and distant speaker detection [7] and separation [11].

13. Baselines

Here, we provide additional implementation details for our *active mapping* baselines for reproducibility (Sec. 5 in main).

- **Random.** At each step t, we generate a random number between 0 and 1 from a uniform distribution. Depending on which quartile of the 0-1 range the random number lies in, we skip visual frames for both egos, sample for just one ego, or sample for both egos.

- **Greedy.** Starting at $t = 2$, we sample visual frames for both egos at every step until we run out of the visual budget B. If the value of B is such that it allows sampling only one visual frame at a certain step (*i.e.* B is odd), we randomly choose the ego for which we sample the frame at that step.

- **Unique-pose.** To implement this baseline, we keep track of the egos' poses during an episode. At any step t, we sample the frame for an ego if it's current pose has never been assumed before by either of the egos in that episode.

14. Architecture and training

Here, we provide our architecture and additional training details for reproducibility. We will release our code.

14.1. Policy architecture

Visual encoder. To encode local occupancy map inputs, our policy π^V (Sec. 4.2 in main) uses a 6-layer CNN consisting of 5 convolutional (conv.) layers followed by an adaptive average pooling layer. The first three conv. layers use a kernel size of 4 and a stride of 2, while the last two conv. layers use a kernel size of 3 and a stride of 1. All conv. layers use a zero padding of 1, except for the third conv. layer, which uses a zero padding of 2. The number of output channels of the conv. layers are [64, 64, 128, 256, 512], respectively. Each convolution is followed by a leaky ReLU [10, 18] activation with a negative slope of 0.2, and a Batch Normalization [6] of $1e^{-5}$. The adaptive average pooling layer reduces the output of the last conv. layer to a feature of size $1 \times 1 \times 512$.

To encode RGB images (Sec. 4.2 in main), π^V uses a separate CNN with 5 conv. layers and an adaptive average pooling layer. Each conv. layer has a kernel size of 4, stride of 2 and zero padding of 1. The number of output channels are [64, 64, 128, 256, 512], respectively. Similar to the occupancy map encoder, each convolution is followed by a leaky ReLU [10, 18] activation with a negative slope of 0.2 and a Batch Normalization [6] of $1e^{-5}$, and the adaptive average pooling layer reduces the output of the last conv. layer to a feature of size $1 \times 1 \times 512$.

We fuse the occupancy and RGB features by concatenating them and passing through a single linear layer that produces a 512-dimensional visual embedding v (Sec. 4.2 in main).

Speech encoder. The speech encoder (Sec. 4.2 in main) in π^V is a CNN with 5 conv. layers and an adaptive average pooling layer. Each conv. layer has a kernel size of 4, stride of 2 and a padding of 1, except for the second conv. layer, which has a kernel size of 8, stride of 4 and padding of 3. The number of channels in the CNN are [64, 64, 128, 256, 512], respectively. Similar to the visual encoder, each conv. layer is followed by a leaky ReLU [10, 18] with a negative slope of 0.2 and a Batch Normalization [6] of $1e^{-5}$. The adaptive average pooling layer reduces the output of the last conv. layer to a feature of size $1 \times 1 \times 512$.

Pose encoder. The pose encoder (Sec. 4.2 in main) in π^V is a single linear layer that takes a normalized pose P (Sec. 4.1 in main) as input and produces a 32-dimensional pose embedding.

Fusion layers. We perform linear fusion of the visual, speech and pose embeddings (Sec. 4.2 and Fig. 2 in main) at two levels. The first level has 4 linear layers and the second level has 1 linear layer. Each linear layer produces a 512-dimensional fused feature as its output.

Policy network. The policy network (Sec. 4.2 in main) comprises a one-layer bidirectional GRU [3] with 512 hidden units. The actor and critic networks consist of one linear layer.

14.2. Mapper architecture

Visual encoder. To encode local occupancy map inputs, our shared mapper f^M (Sec. 4.3 in main) uses a CNN similar to the one used for encoding occupancy maps in π^V (Sec. 14.1), except that it doesn't have a pooling layer at the end. The RGB encoder (Sec. 4.3 in main) in f^M is also similar to the one for π^V, except that it also doesn't have a pooling layer at the end. We fuse the map and RGB features by concatenating them along the channel dimension, and obtain a $4 \times 4 \times 1024$ dimensional feature.

Speech encoder. The speech encoders (Sec. 4.3 in main) in f^M are CNNs with 5 layers that share the architecture with the first 5 conv. layers of the speech encoder in π^V (Sec. 14.1), except that the last conv. layer in both encoders has 1024 output channels.

Modality encoder. For our modality embedding $\hat{m}$ (Sec. 4.3 in main), we maintain a sparse lookup table of 1024-dimensional learnable embeddings, which we index with 0 to retrieve the visual modality embedding ($\hat{m}_V$), 1 to retrieve the modality embedding ($\hat{m}_S$) for the speech from self, and 2 to retrieve the modality embedding ($\hat{m}_{S'}$) for the speech from the other ego.

Occupancy prediction network. The transformer [20] (Sec. 4.3 in main) in our occupancy prediction network comprises 6 encoder and 6 decoder layers, 8 attention heads, an input and output size of 1024, a hidden size of 2048, and ReLU [10, 18] activations. Additionally, we use a dropout [17] of 0.1 in our transformer.

The transpose convolutional network U (Sec. 4.3 in main) consists of 6 layers in total. The first 5 layers are transpose convolutions (conv.) layers. The first 4 transpose conv. layers have a kernel size of 4 and stride of 2, and the last transpose conv. layer has a kernel size of 3 and stride of 1. Each transpose conv. has a padding of 1, ReLU [10, 18] activation and Batch Normalization [6]. The number of the output channels for the transpose conv. layers are [512, 256, 128, 64, 2], respectively. The last layer in U is a sigmoid layer (Sec. 4.3 in main), which outputs the map estimates.

14.3. Parameter initialization

We use the Kaiming-normal [5] weight initialization strategy to initialize the weights of all our network modules, except for the pose encoding layers and fusion layers, which are initialized with Kaiming-uniform [5] initialization, and the policy network, which is initialized using the orthogonal initialization strategy [15]. We switch off biases in all network modules, except for the policy network where we set the biases initially to 0.

14.4. Training hyperparameters.

Policy training. To train our policy π^V using DD-PPO [21] (Sec. 4.4 in main), we weight the action loss by 1.0, value loss by 0.5, and entropy loss by 0.05. We train our policy on 8 Nvidia Tesla V100 SXM2 GPUs with Adam [8], an initial learning rate of $1e^{-4}$ and 8 processes per GPU for 8.064 million policy prediction steps. Among other policy training parameters, we set the clip parameter value to 0.1, number of DD-PPO epochs to 4, number of mini batches to 1, max gradient norm value to 0.5, reward discount factor γ to 0.99, and the value of λ in the generalized advantage estimation [16] formulation for DD-PPO to 0.95.

Mapper training. We train our shared scene mapper f^M (Sec. 4.3 in main) with a binary cross entropy loss (Sec. 4.4 in main) on 4 Nvidia Quadro RTX 6000 GPUs until convergence by using Adam [8], an initial learning rate of $1e^{-4}$ and a batch size of 24.

References

[1] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *ECCV*, 2020. 1, 3

[2] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565, 2015. 3

[3] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 6

[4] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Compute and energy consumption trends in deep learning inference. *arXiv preprint arXiv:2109.05472*, 2021. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 6

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 5, 6

[7] Hao Jiang, Calvin Murdock, and Vamsi Krishna Ithapu. Egocentric deep multi-channel audio-visual active speaker localization. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10534–10542, 2022. 5

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[9] Robert LiKamWa, Zhen Wang, Aaron Carroll, Felix Xiaozhu Lin, and Lin Zhong. Draining our glass: An energy and heat characterization of google glass. In *Proceedings of 5th Asia-Pacific Workshop on Systems*, pages 1–7, 2014. 2

[10] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. Omnipress, 2010. 5, 6

[11] Katharine Patterson, Kevin W. Wilson, Scott Wisdom, and John R. Hershey. Distance-based sound separation. In *INTERSPEECH*, 2022. 5

[12] Senthil Purushwalkam, Sebastia Vicenc Amengual Gari, Vamsi Krishna Ithapu, Carl Schissler, Philip Robinson, Abhinav Gupta, and Kristen Grauman. Audio-visual floorplan reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1183–1192, 2021. 2, 3, 4

[13] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *European Conference on Computer Vision*, pages 400–418. Springer, 2020. 2, 3, 4

[14] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019. 3

[15] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120, 2014. 6

[16] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2016. 6

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 6

[18] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2892–2900, 2015. 5, 6

[19] Ryu Takeda, Yoshiki Kudo, Kazuki Takashima, Yoshifumi Kitamura, and Kazunori Komatani. Unsupervised adaptation of neural networks for discriminative sound source localization with eliminative constraint. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3514–3518, 2018. 3

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 6

[21] Erik Wijmans, Abhishek Kadian, Ari S. Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020. 6