Self-Supervised Incremental Learning for Sound Source Localization in Complex Indoor Environment

Hangxin Liu[1*] Zeyu Zhang[1*] Yixin Zhu[1, 2] Song-Chun Zhu[1, 2]

* Equal Contributors

1. Center for Vision, Cognition, Learning, and Autonomy, UCLA

2. International Center for AI and Robot Autonomy (CARA)

Introduction

Challenges:

- Signal processing approaches use explicit acoustic features, *e.g.* TDOA, IID, incapable in complex environment and non-field-of-view (NFOV) regions.
- Data-driven classification methods have difficulties in collecting training data, and are too cumbersome to adapt in unknown indoor environments.

Our approach:

Given a verbal command from a user, the proposed **incremental learning** framework:

(i) Rank the priority of the rooms to be explored, indicated by the height of the blue bars;

(ii) Explore the rooms following the ranking order;

(iii) A detection of the user leads to a positive labeled sample of the training data on-the-fly.

Advantages:

- Does not require pre-collected data
- Directly appliacble to real-world scenarios without any human supervisions or interventions.

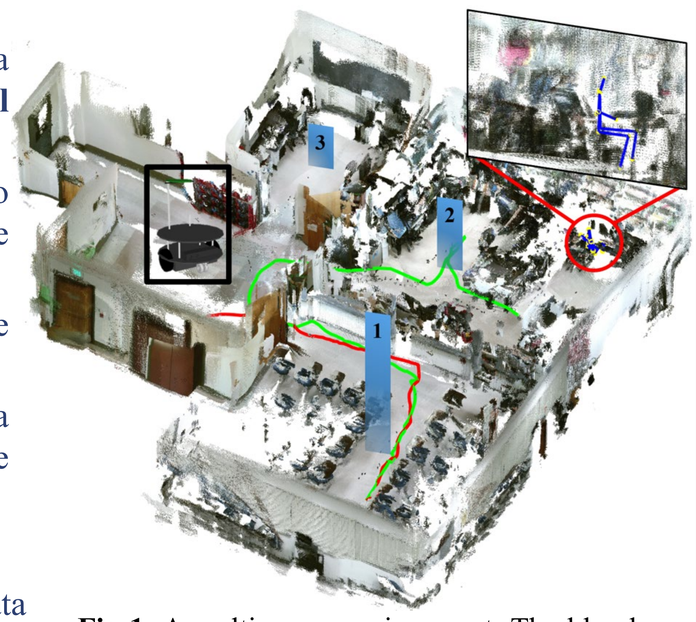
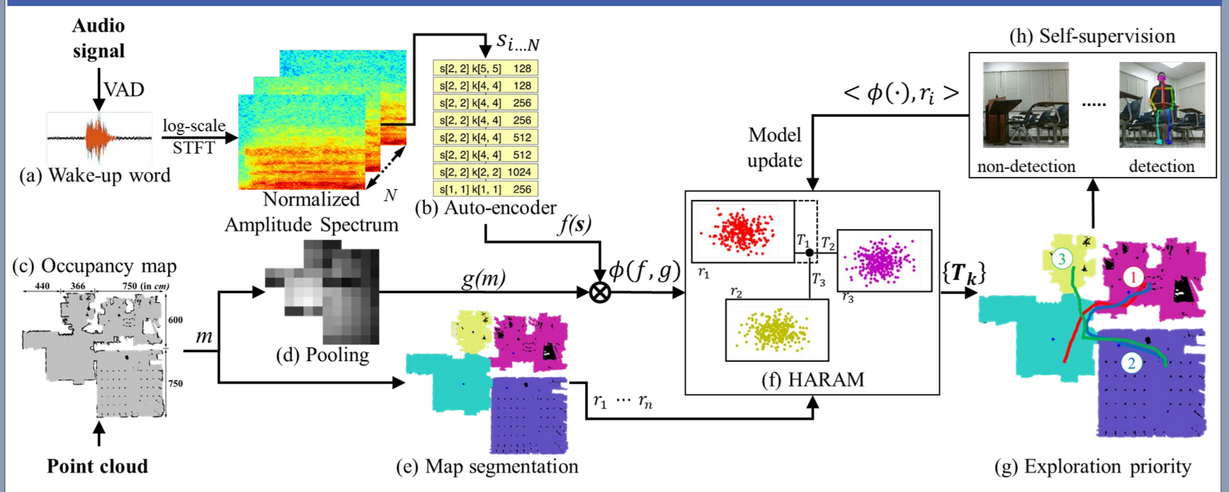


Fig 1. A multi-room environment. The blue bars indicate the exploration priority. The red path shows a wrong exploration whereas the green path shows a subsequent exploration.

Methodology Overview



The proposed approach using a **self-supervised incremental learning** scheme:

(a) The multi-channel signals from the user's wake-up word are picked up by VAD. Each signal is transferred to the amplitude spectrum and normalized to [0, 1];

(b) An auto-encoder is trained to extract implicit features. Each block represents a 2D convolution with stride s[· , ·], kernel size k[· , ·], and the number of channels.

(c) An occupancy map obtained from the reconstructed point cloud.

(d) Down-sampled (c) by pooling and append to (b) to form the feature for learning.

(e) Individual rooms are segmented from the point cloud.

(f) The *HARAM* model is adopted to predict the priority rank of rooms the robot should visit.

(g) The robot self-supervises the learning by exploring the rooms.

(h) The exploration will be labeled as the positive sample if the robot detects the user, which will update *HARAM* model incrementally.

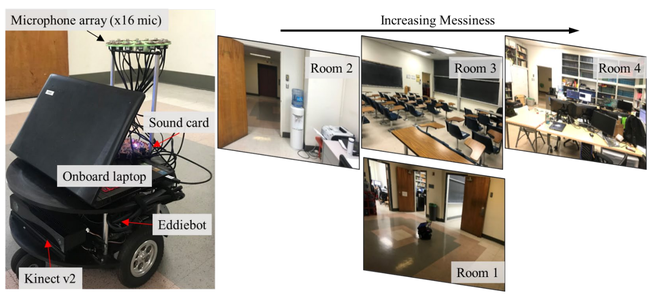


Fig 2. (left) Eddiebot robot setup. A Kinect v2 RGB-D sensor is mounted in the front. A uniform circular microphone array containing 16 microphones is placed on the top. The robot and all the sensors are connected to an on-board laptop that runs the learning algorithm in real-time. (right) A multi-room environment used in experiments. The robot stations in the hallway and the sound sources are in room 1, 2, and 3 with an increasing room complexity.

Open-sourced dependencies:

- Google WebRTC for VAD
- RTAB-Map for SLAM
- OpenPose for human detection
- ROS ipa_room_segmentation

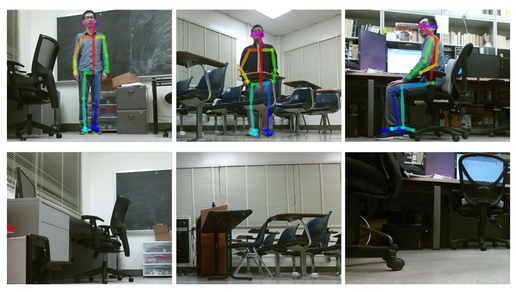


Fig 3. (Top) Examples of the human pose detection. (Bottom) Non-detection examples.

Localization Model By Ranking

Learning: The neural activation function for each room T_k is calculated and the maximal T_* is selected:

$$T_k(\mathbf{\Phi}, \mathcal{R}) = \gamma \frac{|\mathbf{\Phi} \wedge \boldsymbol{\omega}_k^{\phi}|}{\alpha_{\phi} + |\boldsymbol{\omega}_k^{\phi}|} + (1-\gamma) \frac{|\gamma \wedge \boldsymbol{\omega}_k^{r}|}{\alpha_r + |\boldsymbol{\omega}_k^{r}|}, \; T_* = \max\{T_k : k = 1, \dots, R\}$$

where $\mathbf{\Phi} = (\phi, \phi^c)$ is the input features, $\mathcal{R} = (r, r^c)$ is the number of rooms, $\boldsymbol{\omega}_k^{\phi}$ and $\boldsymbol{\omega}_k^{r}$ are their corresponding weight vectors. $\alpha_{\phi} > 0$, $\alpha_r > 0$, and $\gamma \in [0, 1]$ are the learning parameters.

The weight vectors are adjusted incrementally during the learning:

$$\begin{cases} \boldsymbol{\omega}_*^{\phi(\text{new})} = \lambda_{\phi} \left(\mathbf{\Phi} \wedge \boldsymbol{\omega}_*^{\phi(\text{old})} \right) + (1 - \lambda_{\phi}) \boldsymbol{\omega}_*^{\phi(\text{old})} \\ \boldsymbol{\omega}_*^{r(\text{new})} = \lambda_r \left(\mathcal{R} \wedge \boldsymbol{\omega}_*^{r(\text{old})} \right) + (1 - \lambda_r) \boldsymbol{\omega}_*^{r(\text{old})} \end{cases}$$

Ranking: Sorting $\{T_k\}$ by their relative magnitudes. The order of T_k implies the ranking of the candidate rooms based on the current sample received.

Self-supervision via Active Exploration:

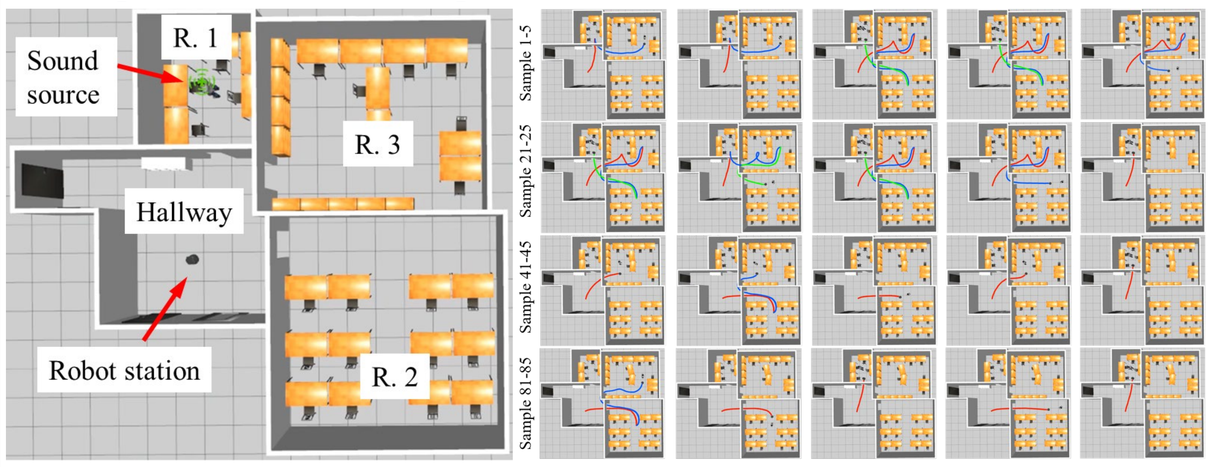


Fig 4. The robot visits the room subsequently following the rank predicted by the model. The red, green, and blue trajectories indicate the first, second, and third rooms the robot visits.

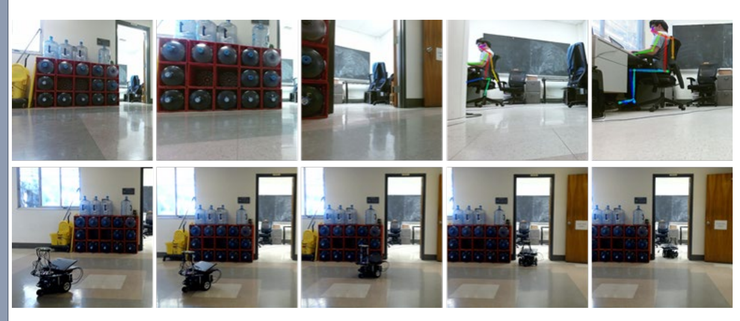


Fig 5. Testing in a physical environment, in which the robot locates the correct sound source with only one visit.

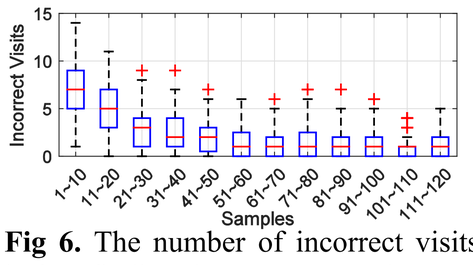


Fig 6. The number of incorrect visits before finding the correct sound source locations in every 10 samples over 100 trails. The number of mistakes decreases rapidly.

Experiments

We compared the proposed method with three baselines:

1. **HARAM+GCC**: Combine HARAM algorithm with GCCPHAT and geometry features;
2. **HARAM+GCCFB**: Add a mel-scale filter bank designed for human voice on top of the GCCPHAT;
3. **MLP + AE**: Use an incremental learning version of the MLP classification method instead of HARAM, and learn from the encoded implicit acoustic feature.

We also investigated the performances using the current 16-microphone setup with only 2, 4, and 8-microphone. Overall, more microphones lead to a better performance with minor fluctuations in the early stage.

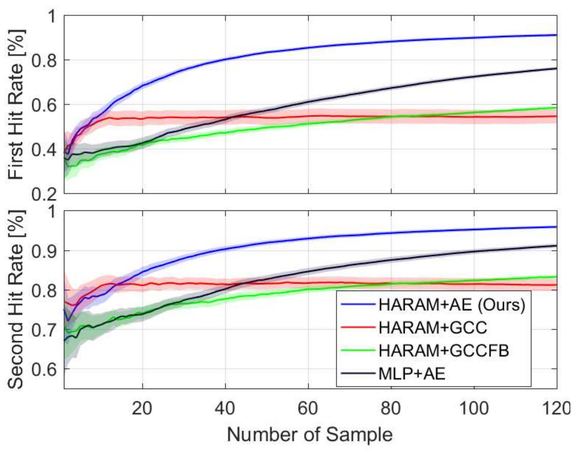


Fig 7. The mean accuracy of (blue) the proposed method and (green and red) two baselines. The first and the second hit rates indicate the robot finds the correct sound source locations within one and two visits, respectively. The color strips indicate the 95% confidence interval over 100 trails.

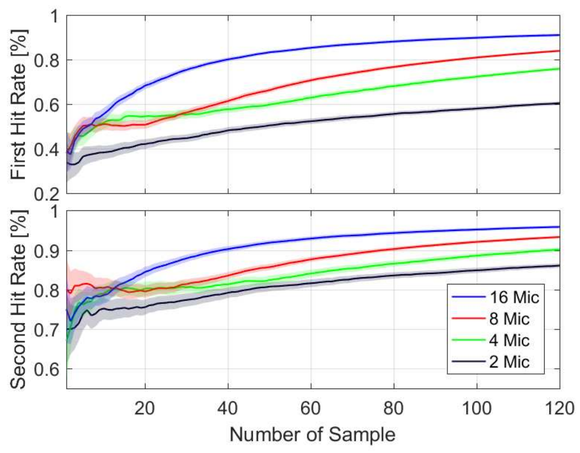


Fig 8. By maintaining uniform microphone placements, we compare the current 16-microphone setup with 2, 4, and 8-microphone setups. The mean accuracy of the proposed method using four different microphone array configurations. The color strips indicates the 95% confidence interval over 100 trails.

The work reported herein was supported by DARPA XAI grant N66001-17-2-4029, ONR MURI grant N00014-16-1-2007, ARO grant W911NF-18-1-0296, and a NVIDIA GPU donation grant.