Acoustic Localization and Voice User Interfaces for Robotic Furniture Applications

John Finberg, *Brown University*

Abstract

Recent advances in Natural Language Processing (NLP) have allowed Voice User Interfaces (VUI) to mature to the point where their integration into everyday objects is becoming inevitable. In the coming years, people will increasingly talk to the things around us, and these everyday objects will react. Furniture, in particular, and the built environment in general, are likely to become sites for inconspicuous, sound-based user interfaces. This prospect offers an opportunity to imagine and consider how sound might be integrated into a range of interfaces. This project demonstrates that a single sensor modality, microphones in this case, can be used across multiple applications, both as VUIs and as a means for localization. These kinds of sound-based interfaces have been integrated into the TableBot prototype, which hides in plain sight as a table until called upon to perform a given task. The prototype validates a model for the integration of voice command functionality that can also perform acoustic localization with the same microphones. When one speaks to this table, it can identify the location of the speaker and can situate itself in relation to the source of the sound command.

Introduction

It is becoming increasingly possible to imagine a world where many of the visual and tactile interfaces with which we are familiar will fade into the background in favor of Voice User Interfaces (VUI). Already, many users are quite comfortable using VUIs that are integrated into AI assistants such as Siri, Alexa, and to a lesser degree, even devices such as the Humane AI pin and the Rabbit r1. [1] However, they have been a trope in science fiction for much longer, including well known examples such as the Star Trek communicator badge, which can be activated with a touch of the finger or by a simple voice prompt. In the years ahead, as this technology matures, we will likely be talking to the objects around us with greater frequency, as complements to, if not outright replacements for, more traditional tactile and visual user interfaces. This possible shift towards VUIs offers interesting opportunities to imagine and consider how sound and voice might be applied to human computer interactions, and the kinds of relationships we might have with these kinds of objects. Sound-based sensing and localization offers an alternative to light-based methods, such as computer vision, LIDAR and other vision-based systems.

For applications that require inconspicuous user interfaces, VUIs and auditory localization allow for designs that use only a microphone (or microphones) as a single, simple input, avoiding the need for multiple sensors (although our prototype uses an array of four microphones, cardinally arranged). In this way, multiple functions can be "piggybacked" onto each other by using microphones in different ways. This system allows these devices to become inconspicuous, with more abstract user interfaces, fading into the background when not called upon to perform a given function.

TableBot: The Robotic Future

Early prototypes of the TableBot design were packed with sensors. LIDAR allowed the first iteration of TableBot to perform Simultaneous Localization and Mapping (SLAM) to localize the robot in a given space. Projectors and cameras allowed for telepresence applications, [2][3] but as the number of features increased, so did TableBot's size and complexity. These early prototypes, like the current prototype, were designed to explore the ways robots might hide in plain sight as part of the built environment, as well as how unseen user interfaces could be called upon when desired to initiate action. Early versions of TableBot could perform a task such as videoconferencing, and then move inconspicuously into the background until needed again. This type of task performance requires that the robot know where it is in space, especially in relation to nearby walls onto which the videoconferencing interface is projected. The current prototype of TableBot builds upon these insights, but simplifies how they are achieved and improves the speed and accuracy with which they are performed.

Careful consideration was given to how these types of objects function as pieces of furniture. TableBot has to work as a practical and stylish piece of furniture before it can take on other functionality. It must be designed to be lightweight, yet robust enough to move about a space, encountering obstacles and people. Not only does it need to function as a robot, but also as a table – a nice table – that meets traditions and conventions of furniture in terms of both aesthetics and function, blending in with a home or an office environment. Tablebot's maple-veneered body and cherry-accented top, as well as its underlying structure, aim to satisfy these concerns. [Figure 1]

Integrating AI and other "smart" technologies into the built environment offers an alternative to the dominant paradigm of personal devices, such as phones, in favor of a model that embraces these technologies as part of the public infrastructure. Just as in years past, when anyone

with a dime could use a phone booth, we envision the integration of telepresence and AI as a kind of public service, available as a part of the built environment. Voice User Interfaces offer promising directions for enhancing this vision.

Figure 1. TableBot front profile

TableBot: The Hardware

TableBot moves using special wheels called Mecanum wheels, which let it glide smoothly in any direction—forward, sideways, or even diagonally—without needing to turn. Its round shape helps make this kind of movement easier and keeps it from looking like it has a "front" or "back," so it blends in better with its surroundings. The robot is compact and sleek, designed to fit just about anywhere.

It runs on a rechargeable battery and uses four small but powerful motors, each connected to a wheel by a belt. The belts help protect the motors from the robot's weight. Everything is held together with strong brackets and smooth bearings to keep the movement steady. [Figure 1]

The robot's brain is a tiny microcontroller called the ESP32, which keeps track of how fast and in what direction each wheel is spinning. It talks wirelessly to a laptop—either nearby or riding on top of the table—using a system called micro-ROS. The laptop sends instructions like "move forward" or "turn left," and the ESP32 sends back updates on how fast it's going and how much battery is left.

For hearing, TableBot has four microphones arranged evenly around its top edge, allowing it to pick up sound from any direction.

Figure 2. TableBot disassembled into its different parts

User Scenario

Consider the following scenario: TableBot is situated within a domestic environment, inconspicuously amongst other furniture. It hides in plain sight, perhaps against the wall or next to a chair, until it is called upon. Perhaps it has a common object or two on it, such as a computer or a book. The user walks into the room and gives the command, "TableBot, come here." Just then, TableBot comes to life, first orienting itself to the direction of the sound and then moving towards that sound. After TableBot arrives within a foot or two of the user, it waits for its next voice command. The user puts down an object on the table, perhaps their drink or another common object, and gives further commands to TableBot. Perhaps the user asks TableBot to do a web search or play a favorite song. TableBot responds. The user then walks over to their desk or to another area of the room, reiterating the command, "TableBot, come to me." TableBot obliges and repeats the prior function, building a map of the space it has traveled from its original position, using only microphones as localization sensors. The user puts down the coffee mug on TableBot, as TableBot awaits the next command.

Evaluation

The design team did a preliminary evaluation of the prototype to validate the functionality of the system and determine the ease of the user experience. The above scenario was repeated with a variety of different distances between TableBot and its target. When the user provided the command to TableBot, the command took between 0.5-3 seconds to process. Once processed, TableBot rotated to the direction of the target, and then moved in the direction of the user, adding another 0.5-1 second delay. A successful run was defined by the table stopping within 0.5 meters of the user. The distance of 0.5 meters was chosen as the error range because it was the reported error of the trained localization model. Thus, localization results should be within that error range if the model has successfully generalized. The system had a success rate of 80% in the 4 to 8 meter range over 10 trials. When tested

at a range of 1 to 3 meters for 35 trials, it had a reported accuracy of 68%.

TableBot's performance in localizing itself in relation to the user performed best within a roughly 10-foot radius. After the user passes beyond this boundary, however, TableBot was unable to accurately predict an angle or distance. This is due to limitations on the training model, being constrained to a 10-meter by 10-meter grid. In future iterations, this space can be expanded. TableBot was most accurate in the 4- to 8-foot range.

In the 1- to 3-meter range, the decrease in accuracy is caused by the choice of experimental microphones. TableBot is equipped with omnidirectional pattern microphones, which are not particularly sensitive to direction. Thus, at a closer distance, the human voice overlaps more with the near microphones. On the other hand, long distances made it challenging for the microphones to pick up. The larger radius also meant that any minor miscalculations in the location angle of the user's voice commands in relation to TableBot began to magnify into larger errors, affecting the accuracy.

With the aim of improving the shorter and longer distance accuracies, some different training techniques were tested. These included time and frequency masking, varying the number of transformer layers in the model, and increasing the training data in our model training. Frequency masking and an increase in training data showed small gains in localization acuarcies. These observations led to considering possible hardware improvements.

Changing the microphone type to the supercardioid pattern will improve the accuracy up close and farther out, offering more directed sound of a greater magnitude. In addition, improvements to the synthetic data generation could be made by generating synthetic audio recordings with more realistic environmental noise scenarios and varied room geometries; these changes might enhance model robustness. In training phases, incorporating audio channel swapping as a data augmentation technique adapted for stereo audio formats may further mitigate the "cone of confusion" effect and enhance localization accuracy. Finally, fine-tuning the model with existing real-world recordings from TableBot will result in an even more accurate localization model.

Objects on the surface of the robot stay mostly stationary as TableBot travels due to its slow acceleration and low center of mass. Items such as books, cups of liquid, and laptops were used in our evaluation. The user can interact successfully with the table, whether removing or adding objects.

Throughout the trials, voice communication with TableBot was noted to be an intuitive and easy means for prompting the robot. In terms of instructions, the user was merely prompted to employ the given command, "TableBot, come here." No confusion was reported by users about how to execute commands. It was interesting to observe the tone of the user when giving commands, which often had an authoritative affect. The affect and tone of the user, not just the actual commands themselves, could be an area of further investigation and might help to improve the usability of the prototype.

Interestingly, during the trials, some users spontaneously issued the "come here" command in their native languages – specifically "ven aquí" in Spanish and "过来这里" in Mandarin. Despite the fact that TableBot was not explicitly programmed to recognize commands in multiple languages, it still responded correctly and navigated towards the speaker. This emergent behavior highlights the potential of this interaction model to lower the barrier of entry, as users are able to engage with the system using natural language without requiring technical jargon or rigid syntax.

Evaluators mentioned that TableBot's familiar form factor, as well as the use of voice commands, contributed to an extremely intuitive interface. Eschewing the need for buried menus or hidden commands, common in most GUIs, evaluators mentioned that they felt a greater sense of immediate control. And while delays produced by processing time, and the relatively slow movement were noted, the overall ease and effectiveness were commented on by the evaluators, and were determined to be comparable to alternative user interfaces. In future work, these features could be sped up or include other features such as variable acceleration and deceleration, to further cue users' commands.

Prior Work

This work draws on research from embedded robotics, voice-based interaction, acoustic localization, and spatial mapping. Prior efforts in these areas have demonstrated the potential for seamless human-robot interaction in indoor environments; however, they often face limitations related to hardware complexity, computational overhead, or reliance on visual sensing. The following sections review key contributions in each domain and highlight opportunities for lightweight, audio-driven alternatives.

Embedded Robotics In Indoor Environments

Gonsher et al. (2020) investigated the integration of robotic functionality into everyday furniture, illustrating how human-computer interfaces can be embedded into built environments with minimal visual intrusion [3]. This laid the groundwork for inconspicuous robotic systems. This concept parallels products such as the Travelmate robotics suitcase, where autonomous behavior is embedded into an otherwise ordinary travel item, allowing users to benefit from robotic assistance without perceiving the object as a robot [17]. Gonsher et al. (2022) extended this vision by incorporating mixed reality and large-screen displays into telepresence robots,

enhancing mediated communication within indoor settings [2].

Complementary efforts in ubiquitous computing and ambient intelligence have explored similar themes, advocating for interfaces that "fade into the background" and support natural interaction within the fabric of everyday life [20]. These systems, however, often rely on visual or tactile modalities, leaving acoustic-based interaction underexplored in the context of embedded robotic systems.

Voice-Driven Interaction And Acoustic Interfaces
Voice assistants such as Amazon Echo [18] or Google Nest [19] have demonstrated the feasibility of natural language interfaces in consumer environments. Despite their accessibility, these systems are typically fixed-location devices that assume a stationary user and lack awareness of spatial context. Some recent approaches aim to incorporate directional audio and user tracking, but they generally require additional sensors or do not scale well to embedded platforms.

In contrast to these efforts, acoustic interfaces based on sound localization offer the potential for spatial awareness without visual input. When integrated into passive or furniture-like platforms, such systems enable more ambient and privacy-preserving interactions.

Sound Source Localization: Classic And Learned Methods
Sound source localization has traditionally relied on digital signal processing techniques such as beamforming or GCC-PHAT to estimate the direction-of-arrival (DOA) of audio signals [8]. Passive sonar approaches [4], allow for localization and tracking without the active emission of signals, making it an appealing property for inconspicuous systems in household settings. However, classical methods often struggle in acoustically complex environments due to reverberation and background noise [16].

To address these challenges, recent studies have proposed machine learning-based approaches, including transformer-based audio encoders [7] and ResNet-Conformer models [10]. These models demonstrate improved robustness to environmental noise but are limited by their reliance on first order ambisonics (FOA) and tetrahedral microphone arrays. Their computational demands also render them impractical for real-time processing on embedded devices [9][11].

Alternate systems using stereo audio and efficient signal processing may serve as a viable compromise, enabling sound-based interaction on resource-constrained hardware.

SLAM And The Constraints Of Vision-Based Mapping
Simultaneous Localization and Mapping (SLAM) is a core technique in mobile robotics, used to build spatial maps and estimate a robot's position within them. Traditional SLAM implementations rely heavily on visual sensors such as RGB-D or LIDAR [22], which provide rich spatial information but introduce privacy concerns and require direct line-of-sight. These systems also demand significant computational resources and may be unsuitable for integration into household furniture or ambient systems.

Acoustic localization, by contrast, can offer spatial awareness without the visual intrusion of cameras or the complexity of 3D mapping pipelines. Systems that use sound cues for positioning and interaction provide a more discreet, lightweight alternative for embedded use in personal environments.

Voice User Interfaces and Passive Auditory Localization of TableBot

Passive Auditory Localization (PAL) is the process by which a source sound can be localized by just its speech or passive sound. The process can be broken down into three steps. The first is the trigger, which is some vocal cue. The next step is identifying the audio sections relating to the cue. Finally, the audio is transformed for processing and run through a pre-trained model to localize the source.

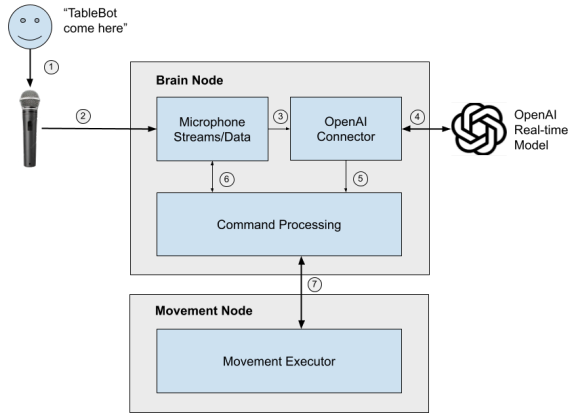

Figure 3. Diagram of the execution flow for a move to command

TableBot's localization processes are organized in the Brain Node. This node is in charge of all the computations performed before passing the computed data off to different nodes for execution of tasks, such as movement [see Figure 3].

The first part of the system streams audio from TableBot's multiple microphones and saves the audio data for future use, organized by timestamp. Before creating a stream, on the robot's device, an aggregate microphone is created

containing all the microphones used for localization. Within the aggregate microphone, the no-drift functionality is activated so that the microphones' clocks do not desync and cause downstream errors with the localization. Using Sounddevice, a Python module, TableBot can stream audio from the aggregate device. This allows TableBot to access each microphone as an individual channel. Based on the microphones' specifications, the streaming is done at 44100 KHz and with a chunk size of 0.25 seconds.

A ring buffer is used to store the audio for later reference. Each audio chunk is stored by its milliseconds from the stream start time. This allows TableBot to retrieve audio based on the audio's recorded time. This functionality will be used later to get only the audio containing the user's speech for localization purposes.

After the audio is collected and stored, the voice command functionality is implemented. An OpenAI real-time model handles the voice and language processing system. Instead of transcribing the audio and sending the text to a typical LLM, the streamed audio is sent straight to the real-time model. The model is given a prompt about its status as an autonomous table and a series of possible actions it can take. It is asked to return the name of the command it is being asked to perform if there is a valid command, and if not, to do nothing. The audio is sent every chunk via a WebSocket to the OpenAI servers. A receiver thread is spun up to parse the responses from OpenAI. This thread continues to pull messages from the socket to keep track of any commands. In addition to parsing commands, OpenAI gives messages for the beginning and ending of speech, which include when that event happened in terms of the time from the start of the stream in milliseconds. The speech events are stored in a queue, where an event has both a start and stop event with respective time markings. The start event is added, and then the end event is added to the last event without an end event. When a command is received, it is then attached to the first completed speech event in the queue.

The next step is for TableBot to process the command. First, it is determined what logic needs to be executed based on the given command. Commands sit in a conditional list where TableBot checks to see if the received command exists and then executes the associated logic. Adding a command is as simple as expanding this conditional with the new command name, as well as adding the new command name to the robot's prompt of possible actions. When a move command is received, TableBot first grabs the audio data for the speech event and then passes the audio to the localization system. Finally, it sends the angle and distance to the movement system.

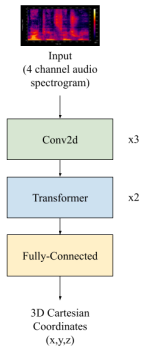

Figure 4. Diagram of PAL model

The localization system is powered by a custom-trained deep neural network consisting of approximately 1.7 million trainable parameters. The model takes in a sound spectrogram from the microphones and then produces a coordinate on the 3D Cartesian coordinate grid of where it predicts the sound to have originated. The first step is converting the audio data to a spectrogram. This is handled by the spectrogram function in the scipy.signal module. Next, the spectrograms are passed into the network. The first three layers of the network are 2D convolution layers. Each convolution block H employs a 3 x 3 kernel with a stride size of 1, followed by ReLU activations and batch normalization. These layers apply a series of filters to the input spectrograms, enabling the network to detect and extract low-level features such as edges, textures, and small-scale temporal and frequency patterns. After each convolution block, a 2 x 2 max pooling operation is applied, which downsamples the feature maps by selecting the maximum value in each non-overlapping region. By focusing on the local regions of the spectrograms, these convolutional layers effectively learn how to represent fine-grained details that are essential for subsequent processing.

$$H = MaxPool(ReLU(BatchNorm(Conv2D(X))))$$

Following the convolutional layers, the architecture incorporates two multi-headed self-attention transformer layers. Each transformer layer contains 4 attention heads, a hidden size of 64, and feed-forward network dimensions of 64. Unlike convolutional layers, which have a fixed receptive field, transformer layers leverage self-attention mechanisms to evaluate relationships between all parts of the input simultaneously. This ability allows the network to capture global dependencies and context across the entire spectrogram. Moreover, the inclusion of positional encodings in these layers enables the model to understand the order and relative positioning of features, thereby embedding the spatial structure of the spectrogram into its representations. This dual capability of capturing both

long-range interactions and positional information is vital for tasks where the global context influences model performance.

Experimental ablation studies further revealed that using a small number of transformer layers, specifically two, yielded better performance compared to architectures with a greater number of layers. The results indicated that beyond a certain point, additional transformer layers did not translate into proportional improvements in model performance. It is important to note that these ablations were conducted under the constraints of training on a single NVIDIA RTX3060 GPU, which limited the scope of experimentation.

Previous studies have shown that integrating convolutional operations with self-attention mechanisms can enhance feature extraction beyond what is achievable when these methods are used independently [14]. The convolutional layers efficiently capture local patterns, while transformer layers provide a complementary global perspective. This combination ensures that the model benefits from both detailed, localized information and an understanding of the broader context within the spectrogram.

The final part of the network consists of three linear layers [see Figure 4]. These layers function as a mapping mechanism, translating the high-dimensional feature representations obtained from the preceding layers into a coordinate space. A ReLU activation follows each linear layer. To improve generalization and prevent overfitting, dropout with a rate of 0.1 is also applied after each linear layer. The linear layers essentially aggregate and refine the features to produce the final output. This step is critical for ensuring that the abstract features learned by the network are transformed into actionable predictions.

The model was trained using synthetic data generated with approximately the same microphone setup as the robot. The recordings were generated using Pyroomacoustics [12], and over 200,000 recordings were collected for training. The simulation was set up to match TableBot's structure and microphones as best as possible. Thus, a circular planar array of four microphones was replicated with the microphones operating in a cardioid pattern. The microphones were positioned 0.25 m apart from each other, and the array was placed in the center of simulated rooms. The microphones were also rotated to their correct orientation. Thus, a microphone on the left would be facing west and a microphone at the bottom would be facing south. Each room was limited to a shoe box-shaped 10 m x 10 m x 10 m.

A series of different AI voices were recorded saying different phrases with differing tones. These voices were sourced from ElevenLabs.

Finally, the following algorithm was run 200,000 times.

1. Create a room
2. Initialize the microphone array
3. Select a random AI voice and place it randomly within the room
4. Create random environmental noise (1-3 times per room)
    a. Initialize a white noise source in the magnitude of 0.001-0.01
    b. Randomly place the noise source in the room
5. Run the simulation and record the signals from the microphone array
6. Create spectrograms from the microphones and save them

The dataset was over 100 GB in size and thus required creating a custom data loader to load only the active training and testing data. The entire dataset was split into 80% training and 20% testing sets. The model was trained for 50 epochs using a learning rate of 1e-4 and a batch size of 32, with early stopping applied after five epochs of no improvement. To enhance generalizability, time and frequency masking augmentations were applied to the training set [9]. Experimental ablation studies demonstrate that incorporating time and frequency masking enhances model performance. The model was tested using mean squared error (MSE) loss, a standard evaluation metric for predicting 3D Cartesian coordinates in the sound source localization literature.

Finally, in the case of a move command, comes the process of instructing the robot to move. Once the coordinates have been localized, they can be passed to the movement node for execution.

The movement node is a separate service that is fixed inside TableBot. The node is a ROS2 service containing a websocket, which enables the Brain Node to control the robot's ROS2 components. When TableBot receives instructions, they come in the form of a distance (in meters) and an angle (in degrees) through a WebSocket connection. The system then converts these instructions into timing, essentially calculating how long TableBot needs to move at fixed speeds to achieve the desired movement. TableBot always moves in two distinct phases: first rotating, then moving forward. For rotation, it uses a fixed speed of 0.5 radians per second, and for forward movement, it moves at 0.3 meters per second. The sequence includes built-in delays: a 2-second wait before starting any rotation, and a 1-second buffer between completing the rotation and starting the forward movement. For example, if TableBot was asked to move forward 1 meter after turning 90 degrees, it would first wait 2 seconds, then spend about 3.2 seconds turning,

wait another second, and finally move forward for about 3.3 seconds before stopping. The entire movement is controlled by precisely timing when to apply these fixed speeds, rather than varying the speeds themselves.

Discussion

Practical evaluations indicated high localization accuracy, correctly predicting the general direction of the speaker. However, the precision of these predictions varied significantly. Importantly, for usability and safety reasons, accurate distance estimation emerged as a critical factor. Inaccurate distance estimations, especially overestimations, pose potential safety risks by causing the robot to approach too closely or collide with the user.

Additionally, the use of stereo microphone configurations significantly degrades localization accuracy due to the psychoacoustic "cone of confusion" effect [1]. Although this issue can potentially be ameliorated through data augmentation techniques such as audio channel swapping [1]. However, current methods for channel swapping are primarily designed for FOA or tetrahedral audio formats [9]. These methods would require adaptation to be effectively applied to stereo audio setups.

In the TableBots current implementation of the voice and command processing units, there is a list of commands with static logic written for each command. As discussed above in the technical implementation, adding to these commands is easy. All that is required is modifying the robot's prompt with the new command and providing the logic in the command's conditional. Commands have access to the audio data and the returned text from OpenAI.

While commands currently do not take in parameters from the user's speech, there are some scenarios where this might be useful. Imagine that one has a full cup of water on the TableBot, and would thus like the table to move very slowly, so as not to knock it over. A user might want to say, "Come over here, very slowly." In the bot's current form, it does not extract parameters for commands. However, this would be achievable with simple prompting and parsing of the OpenAI response. For each command, a developer could specify optional parameters for OpenAI to extract from a user's request, such as speed.

It may even be possible for these parameters to be naturally inferred. Imagine a user asking, "Bring me my cup of tea." The chat model could assume that, if it is to carry tea, it should automatically flag a slow parameter for its movement.

Many other scenarios and user groups can be imagined beyond our initial demonstration. For example, seniors and other populations living with mobility issues, especially those who use assistive mobility devices such as canes, walkers, and wheelchairs, might find a near-proximity attendant table surface helpful. Such a table surface could follow the user through a space, helping to carry objects from one place to another when the user is unable to do so. These objects could also be called upon as needed. TableBot's potential as an assistive technology device – one that allows seniors to "age in place," for example – could be supported in future work.

To handle more advanced workflows, the commands could be altered to accommodate an agent structure, where each command becomes a tool that the chat model can invoke to achieve the user's request. Imagine a user asks, "Can you follow me while I talk with someone else?" The TableBot would have the tools Move and Localize, where Move would move the robot to a specific coordinate, and Localize would localize the current sound and provide its coordinates. With these tools, the agent would realize it needed to call Localize and Move repeatedly to follow the user as they talk to their companion. Another example where tooling flexibility opens new possibilities is when a user says, "Remember the coffee machine is here." The robot could then localize the audio using the Localize tool and note the position in memory. By tooling an agent with the robot's physical capabilities, there are new infinite combinations of tools that can solve more advanced user requests.

This prototype validates a model for the integration of a sound-based user interface with a sound-based localization system, using the same microphones for both tasks. The evaluation and testing have shown that VIL can be a more natural and accessible interaction for users that allows them to interact with a technical robot without having to learn a specific language or pattern for interaction.

In future work, the hope is to build upon these initial innovations and develop more robust approaches to what can be referred to as VIL systems or Voice Interface with Localization. One can imagine at least three different approaches to the further development of VILs.

The primary area of interest is in using VILs for more robust mapping, more akin to SLAM mapping. This involves storing data in memory and giving the user the ability to command the robot to remember (or forget) particular positions in space. In this way, the user and the robot collaborate to create a map of the space, based on the positional cues given by the user.

In addition to Passive Acoustic Localization, we are interested in integrating Active Acoustic Localization to the VIL system, adding a third microphone-based application to the system. This will allow the robot to send out a signal that can bounce off obstacles in the space, adding object avoidance to Passive Acoustic Localization, which currently only works in relation to voice commands, but not physical obstacles.

Finally, where appropriate, one can envision VILs being complemented by other sensor systems, including vision-based systems, and more traditional systems such as SLAM. The integration of these other approaches to VILs can be developed to address a broader range of applications.

Conclusion

Imagine a world where everyday things come to life when spoken to. Like the Sorcerer's Apprentice, who could give life to quotidian objects simply by reciting a spell, Voice Interfaces with Localization (VIL) validate a new model for enhanced Voice User Interfaces (VUI) by integrating verbal commands with acoustic localization [21]. This is accomplished with the economy of a single sensor: a microphone. One can imagine near-future scenarios where furniture equipped with these VIL systems hides in plain sight until they are called upon to perform a task, contributing to a paradigm where robotics, AI, and other emerging technologies are integrated into the built environment, both as public infrastructure and as domestic furniture.

As the potential for these technologies grows, it is worth considering the kinds of relationships they foster between humans and technology. One can imagine a cooperative, collaborative relationship emerging, akin to an attendant service animal, for instance. Or, one can also imagine a non-reciprocal relationship emerge, setting up a Master-Slave dialectic, where machine and human become unequal, yet interdependent upon each other. Speech has almost mystical power when projected onto objects. Speaking to objects is much closer to a human-to-human interaction than a keyboard or mouse can provide, for example. So, it is worth considering how interfaces such as VILs might foster healthier relationships between humans and technology in the coming years. As the Sorcerer's Apprentice learned, speech is powerful, especially when commanding everyday things.

Acknowledgements

Many people have made significant contributions to this project. Professor Gonsher's guidance, insights, feedback throughout this past year have been invaluable. Professor Huang provided very useful comments on the paper drafts. Collaborating with Nicolas Perez on the localization was very helpful. The prior work of Jennifer Tran and Jipuwapt Mokkamakkul helped me clarify my approach to this project. Joshua Phelps and Siddharth Diwan laid the groundwork for the physical robot. Thank you to my mom, Melanie Piech, and Olivia Suomi for their edits and support.

Refrences

[1] Shreyas Sen and Arunashish Datta. 2024. Invited: Human-Inspired Distributed Wearable AI. In Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC '24). Association for Computing Machinery, New York, NY, USA, Article 364, 1–4. https://doi.org/10.1145/3649329.3663513

[2] Gonsher, I., Han, Y., Desingh, K., & Gokaslan, A. (2022). Prototyping mixed reality large screen mobile telepresence robots. In *5th International Workshop on Virtual, Augmented, and Mixed Reality for HRI*.

[3] Gonsher, I., & Kim, J. Y. (2020, March). Robots as furniture, integrating human-computer interfaces into the built environment. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 215-217). https://doi.org/10.1145/3371382.3378235

[4] L. Mattos and E. Grant, "Passive sonar applications: target tracking and navigation of an autonomous robot," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, New Orleans, LA, USA, 2004, pp. 4265-4270 Vol.5, doi: 10.1109/ROBOT.2004.1302388.

[5] S. Macenski, A. Soragna, M. Carroll, Z. Ge, "Impact of ROS 2 Node Composition in Robotic Systems", IEEE Robotics and Autonomous Letters (RA-L), 2023.

[6] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics vol. 7, May 2022. https://docs.ros.org/en/iron/Citations.html

[7] Zheng, Z., Peng, P., Ma, Z., Chen, X., Choi, E., & Harwath, D. (2024). Bat: Learning to reason about spatial sounds with large language models. *arXiv preprint arXiv:2402.01591*.

[8] Valin, J. M., Michaud, F., Hadjou, B., & Rouat, J. (2004, April). Localization of simultaneous moving sound sources for mobile robot using a frequency-domain steered beamformer approach. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004* (Vol. 1, pp. 1033-1038). IEEE.

[9] Wang, Q., Du, J., Wu, H. X., Pan, J., Ma, F., & Lee, C. H. (2023). A four-stage data augmentation approach to resnet-conformer based acoustic modeling for sound event localization and detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *31*, 1251-1264.

[10] Wang, Q., Dong, Y., Hong, H., Wei, R., Hu, M., Cheng, S., Jiang, Y., Cai, M., Fang, X., & Du, J. (2024). *THE NERC-SLIP SYSTEM FOR SOUND EVENT LOCALIZATION AND DETECTION WITH SOURCE DISTANCE ESTIMATION OF DCASE*

2024 CHALLENGE [White paper]. DCASE2024 Challenge.

[11] Wilkins, J., Fuentes, M., Bondi, L., Ghaffarzadegan, S., Abavisani, A., & Bello, J. P. (2023). Two vs. four-channel sound event localization and detection. *arXiv preprint arXiv:2309.13343*.

[12] Scheibler, R., Bezzam, E., & Dokmanić, I. (2018, April). Pyroomacoustics: A python package for audio room simulation and array processing algorithms. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 351-355). IEEE.

[13] Prerak Srivastava. Realism in virtually supervised learning for acoustic room characterization and sound source localization. Machine Learning [cs.LG]. Université de Lorraine, 2023. English. ⟨NNT : 2023LORR0184⟩. ⟨tel-04313405⟩

[14] Bello, I., Zoph, B., Vaswani, A., Shlens, J., & Le, Q. V. (2019). Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 3286-3295).

[15] Goethe, J. W. von. (1797). *The Sorcerer's Apprentice* (E. Zeydel, Trans.). Beezone Library. Retrieved from https://beezone.com/the-sorcerers-apprentice

[16] H. Lim, I. -C. Yoo, Y. Cho and D. Yook, "Speaker localization in noisy environments using steered response voice power," in IEEE Transactions on Consumer Electronics, vol. 61, no. 1, pp. 112-118, February 2015, doi: 10.1109/TCE.2015.7064118.

[17] Travelmate Robotics. (2019). *Travelmate: Fully autonomous suitcase and robot companion. https://wefunder.com/travelmaterobotics*

[18] Amazon (2015). *Amazon Echo Dot. https://www.amazon.com/Amazon-vibrant-helpful-routines-Charcoal/dp/B09B8V1LZ3?th=1*

[19] Google (2019). *Google Nest Mini. https://store.google.com/us/product/nest_audio?hl=en-US*

[20] Weiser, M. (1994). *Creating the invisible interface* [Invited talk]. *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology* (pp. 1). Association for Computing Machinery. https://doi.org/10.1145/192426.192428

[21] "The Sorcerer's Apprentice By Goethe – Beezone Library," *beezone.com*. https://beezone.com/the-sorcerers-apprentice

[22] D. Watkins-Valls, J. Xu, N. Waytowich and P. Allen, "Learning Your Way Without Map or Compass: Panoramic Target Driven Visual Navigation," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 5816-5823, doi: 10.1109/IROS45743.2020.9341511.