IoT Sonic Localization on Commodity Hardware

Clifford Bakalian, Justin Goodman

University of Maryland, College Park

ABSTRACT

One key aspect of multi-agent tasks is sharing and maintaining agent location data. Typical approaches to localization fail for constrained domains. GPS fails for indoor problem domains, and sophisticated sensors fail for limited-budget domains. In this paper, we explore the viability of using sound to localize agents relative to each other. We constrain ourselves to commodity hardware with a max budget of $60. We show that, theoretically, any agents ($n \geq 2$) can localize themselves relative to each other. Our proof-of-concept robots provide valuable insight towards solving this problem.

1. INTRODUCTION

Localization is the problem of figuring out where an agent is located relative to some reference frame. Agent localization is the first step to solving most distributed-agent problems. For example, distributed mapping of an unknown environment is an important robotics problem requiring agent localization. Many approaches to localization exist, each with pros and cons. GPS allows localization of agents to latitude/longitude Earth coordinates, yet must be performed outside. Indoor fingerprinting techniques allow localization to an established beacon grid, yet requires existing infrastructure and consistent maintenance. SLAM techniques remove the fingerprinting requirement, yet typically place other environmental restrictions. Finally, most localization techniques assume the agent has access to accurate measurements from many sensors. We explore solving localization under the following constraints:

1. minor environment assumptions
2. limited hardware capacity

In this paper, we contribute a proof-of-concept robot-server architecture to facilitate the localization first-step for a general robot mapping problem. Our localization procedure, **SLOMO** (Sonic LOcalization using MOtion), uses audio signals passed between agents to derive distance estimates, which are used for multilateration to derive relative positioning. Our robots, which we call **SLOB**s (Sonic LOcalization Bot) in this paper, are built using cheap commodity hardware. Our SLOBs feature a two-board communication architecture that includes wireless internet connectivity, two-axis motion functionality, audio source functionality, and sensor array accessibility including two microphones, an accelerometer, a gyroscope, and an ultrasonic sensor. Each SLOB costs about $58 to build – we list our parts in table 1 in section 3. Our code is also available open-source on GitHub [1].

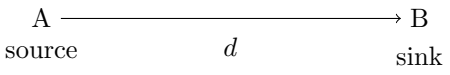
The rest of the paper is organized as follows. Section 2 details the theoretical backing behind sonic localization. Section 3 outlines our implementation and reproducibility information. Section 4 discusses our proof-of-concept results. Section 5 concludes the paper with a discussion of limitations and future work.

2. THEORY

In this section, we evaluate the theoretical viability of such a sonic system.

2.1 Distance Estimation

To some degree of error, the speed of sound in air is constant at $c = 343$ meters per second. We can use this to estimate distance between a source and sink node. To do this, we must ensure our speaker and listener are talking and recording at the exact same time. Given this, then distance can be measured easily.



Let T be the number of seconds at which the sink detects the source's signal. Then, in meters,

$$d = T \cdot c$$

It is not apparently obvious how to find T, however. In noisy environments, the source's signal may appear undetectable at the sink by human ear. Yet, it is possible to detect the signal using cross-correlation, given the source signal. Assume that the sink records for longer than the source's signal. Let S_{src} denote the source's sent signal, and let S_{rec} denote the recorded sink signal. Assume the sample rates of both signals are the same – if they are not, then interpolate S_{src} until the sample rate is the same. Then the cross-correlation is given by element-wise multiplying the Fourier transform of the received signal with the complex conjugate of the

Fourier transform (reversing) of the sent signal, and taking the inverse Fourier transform of the result. The maximum-likelihood index at which the source signal appears in the received signal is then the index that produces the maximum value in the aforementioned convolution.

$$R = F^{-1}\left(F(S_{\text{rec}}) \odot \overline{F(S_{\text{src}})}\right)$$
$$i^* = \underset{i}{\text{argmax}}\ R_i$$

In practice, this requires zero-padding and offset corrections. We omit these details for brevity. Then i^* is the most likely sample index at which S_{src} appears in S_{rec}. Let N be the sink's sample rate in samples per second (hertz). Then the time associated with index i^* is given by

$$T = i^* \cdot \frac{1}{N}$$

which means the distance d, in meters, between the source and sink is

$$d = \frac{i^*}{N} \cdot c$$

Note that given the sampling rate, we are able to calculate the distance traveled per sample, which gives us a measure of distance detection *fidelity*. For the distance traveled per sample, we calculate

$$\Delta d = \frac{(i+1)\cdot c}{N} - \frac{i \cdot c}{N} = (i + 1 - i)\frac{c}{N} = \frac{c}{N}$$

This means fidelity and sample rate are inversely related – as our sample rate increases, our change in distance between samples decreases (meaning, our fidelity gets *better*, more fine-grained).

Given a required fidelity, we can calculate the minimum required audio sampling rate:

$$N = \frac{c}{\Delta d}$$

For meter-level GPS accuracy, our system must support a minimum sample rate of $N = 343$ samples per second. As our scale reduces, the sampling rate increases. An ideal target is centimeter-level accuracy, which requires a minimum sample rate of $N = \frac{343}{0.001} = 34.3$kHz. Considering modern microphones can record at 44.1kHz, minimum, 34.3kHz seems like a plausible target.

2.2 Multilateration

For a general k-dimensional Euclidean space, we need at least $k + 1$ known-location base stations, and distances to said stations, to $(k + 1)$-laterate an unknown source. For locating a point in 2D space, one known-location sink is not enough. In fact, we need at least three base stations with known locations to pinpoint the precise unknown location. Figure 1 showcases this idea. The three microphones m_1, m_2, m_3 each hear a corresponding distance d_1, d_2, d_3 from the source. If the microphone locations are known, and the distance calculations are without error, then we can calculate the source location (figure 1a). Let (x_i, y_i) denote the position of m_i. Let (x, y) be the unknown position of src. Then we can solve the resulting system for x and y:

$$d_1^2 = (x - x_1)^2 + (y - y_1)^2$$
$$d_2^2 = (x - x_2)^2 + (y - y_2)^2$$
$$d_3^2 = (x - x_3)^2 + (y - y_3)^2$$

Notice that this system may have no real solution. Such a situation arises when the calculated distances are within some amount of error such that not all circles are overlapping. Given some distance error, we might obtain resulting distances with circle radii as in figure 1b.

In our domain, we incur distance error if the source and sinks are not properly synchronized. For example, if we have a sample rate of $N = 1000$ samples per second, then a synchronization offset of 10 milliseconds corresponds to

$$\Delta d = (0.01 \cdot 1000) \cdot \frac{343}{1000} = 3.43\text{m}$$

Notice that this does not depend on the sampling rate. The distance error solely depends on the synchronization. To achieve the target centimeter-level accuracy, we require a minimum synchronization offset of $\frac{0.01}{343} \cdot 1000 = 0.002915$ milliseconds. This may be infeasible for a distributed system connected via the internet; it seems modern methods can only achieve 10 millisecond accuracy [2]. This also massively dwarfs the sampling rate requirement, which implies time synchronization will be our most significant source of error. As we discuss later, this is indeed the case.

2.3 Localization

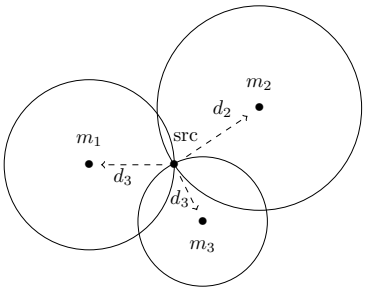
By combining the previous ideas, we can realize requirements for a distributed mobile localization system:

1. more than two sinks with known locations
2. high enough sampling rate
3. low enough synchronization offset

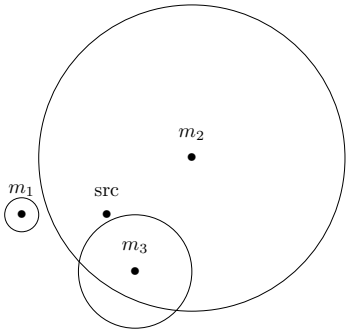
Let us assume theoretically we have a feasible sampling rate and synchronization. Then, we need some way to satisfy known-location sinks. We present two viable options to solve this.

2.3.1 With Motion

Our first localization technique, **SLOMO** (Sonic LOcalization using MOtion), is as follows. Consider two agents, each equipped with two attached microphones.



(a) With zero error, trilateration is successful.



(b) If distance estimates incur error, trilateration may fail to identify three intersecting points.

Figure 1: Multilateration in 2D space (trilateration) succeeding (1a) and failing (1b).

Further assume one agent a_0 is mobile and can accurately move forward d meters and return $-d$ meters. Then we can calculate the location of the other agent a_1 with the following procedure:

1. a_1 sends audio source signal, and a_0 listens for signal at both microphone sinks
2. a_0 moves forward d meters
3. a_1 transmits audio signal to a_0 again
4. a_0 moves backwards $-d$ meters

We then calculate the distances from each microphone to the source as before. We induce a_0's starting position to be the origin $(0, 0)$ of some Cartesian global plane. Given the microphone offsets to the agent's center, and the travel distance offset d, we now have four known-location base stations (the microphone positions). Then we perform multilateration to obtain the position of a_1 *relative to* a_0. Figure 2 shows this idea.

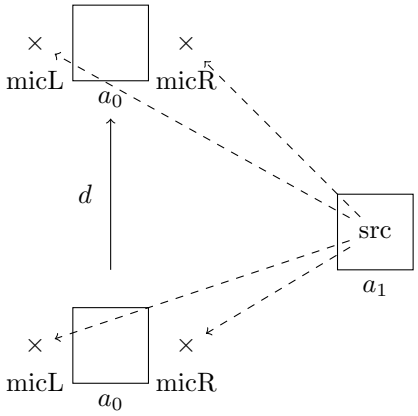


Figure 2: SLOMO between a mobile leader listener and a stationary secondary speaker. The leader microphones are spaced out for clarity, but would be physically located on the agent in a similar configuration.

This procedure is easily abstracted to $n > 2$ agents:

1. designate one agent a_0 as the leader
2. for each agent a_i that is not the leader, localize a_i to a_0 according to the previous procedure

We use this approach for our proof-of-concept.

2.3.2 *With Deduction*

While we did not look into it for this paper, we hypothesize another viable localization method. We denote this approach as **SLODE** (SOnic LOcalization using DEduction). For three agents a_0, a_1, a_2, each with two microphones and a speaker, do the following:

1. designate a_0 as the leader (origin)
2. calculate source to left-right microphone distances between each pair
3. deductively resolve the correct source point by exploiting the distances and induced left-right source directions from each microphone pair

Calculating distance from estimated motion vectors can yield error in SLOMO. As such, SLODE may be better due to the lack of motion assumption. We leave the theoretical, and empirical, analysis of SLODE to future work.

3. IMPLEMENTATION

We realize our SLOBs using commodity hardware. Our hardware choices are presented in table 1. Our robots cost about $58, which we believe is well within reason for such a project. Step-by-step reproducibility information is available in our GitHub repository [1].

Our SLOBs contain two micro-controllers (Arduino Nano and ESP8266), which communicate using serial transmit and receive communication lines. We use the Arduino Serial software interface to facilitate this. Due to wiring and sensor constraints, and analog microphone requirements, we placed both microphones on

#	Part
1	ESP8266 NodeMCU
1	Arduino Nano
1	HC-SR04 Ultrasonic Sensor
2	KY-038 Microphone
1	PAM8403 5V Amplifier
1	4Ω 3W Speaker
1	GY-521 MPU-6050 Module
1	L9110S H-Bridge
4	3-6V Motor
1	DIY Robot Chasis
1	Toggle Switch
1	9V 600mAh Li-ion Rechargeable battery

Table 1: Parts list. Note that our robot chassis included the four DC motors, as well as mounting hardware. Note that our Wi-Fi board is an entire micro-controller made by HiLetgo (ESP8266 CP2102 ESP-12E). Note that we use the LAFVIN Nano with Atmel Atmega328P-AU micro-controller unit and CH340G USB chip. We omit our specific wire and breadboard counts, but our most recent wiring schematic is available in our GitHub repository [1].

our Nano board. This unfortunately limits our sampling rate to the maximum baud possible through the serial communication channel. The Serial library sets the maximum baud at 115200 bits per second. Our microphones produce analog signal amplitudes between 0 and 1023, requiring 10 bits per sample. Under alignment, this is 2 bytes (16 bits) per sample. Each SLOB has two microphones, so we must collect and transmit at least 20 bits per sample. Under alignment, this is 4 bytes (32 bits) per sample. Given the maximum baud, we can theoretically transmit $115200 \cdot \frac{1}{32} = 3600$ left-right samples per second. We are also limited by our micro-controller clock speed. The Arduino Nano performs our audio sampling, and clocks at 16MHz. Assume, liberally, that it takes 1000 clock cycles to record one sample. Then theoretically a Nano board can record audio at $16000000 \cdot \frac{1}{1000} = 16000$ samples per second, or a 16kHz sampling rate. It follows that our serial communication severely restricts our audio sampling performance. At the maximum transfer rate, though, a 3.6kHz left-right sampling rate achieves a single-microphone sampling rate of 1.8kHz yielding a theoretical 19cm distance fidelity.

Our software implementation is available on GitHub [1]. We implemented SLOMO using a server written in Golang. An AMD Ryzen 7 3700X CPU computer hosts the server and LAN. Each SLOB contains a Wi-Fi micro-controller which provides client and server functionality. This allows our SLOBs to maintain a bidirectional communication channel with our server. In this way, our SLOBs act as APIs for which the server interacts. The server sends commands – SPEAK, LISTEN, MOVE, and SENSOR – via HTTP POST requests to an individual SLOB, and our SLOBs send result information – timing information, audio samples, move results, and sensor readings – via HTTP POST requests back to the server. We list our full architecture in figure 3.

Each SLOB is about 25cm × 15cm. The SLOB's speaker is placed, facing upright, as close to the center as possible. The microphones are placed upright in the center close to the outermost edges of the car. The microphones are about 10 centimeters apart.

At the start, the server waits for all n robots to register. Registration is performed by a SLOB making a POST request to the server after the SLOB turns on and finishes its set-up procedure. On registration, the SLOB tells the server its IP address and its millisecond clock time. The server assigns a unique then assigns a unique ID (counting up from zero) to the SLOB, then saves this clock time to use as an offset for localization timing. The server replies to the registrant with the ID. The SLOB saves this ID, and sends it with all messages sent to the server henceforth.

Localization starts after all SLOBs are registered. The SLOB with ID 0 is designated as the leader, and all SLOBs are localized to the leader using the SLOMO technique described earlier. LISTEN and SPEAK requests are sent to the localizing SLOBs with a tuned *delay time* to accommodate time synchronization. The localizing bots will wait for their calculated delay time, then actually listen and speak. The listener samples from its left and right microphones as fast as it can, while the speaker plays a continuous 300Hz tone. The listener POSTs to the server with an array of integers representing the left and right sampled microphone amplitudes, as well as its start time and total time of recording (used for calculating the audio sampling rate). The speaker POSTs to the server its start and total playing time. The server receives these requests, tells the leader SLOB to move forward 100 centimeters, repeats the previous process, then tells the leader SLOB to move backwards 100 centimeters.

Once two bots complete the localization procedure, the server performs multilateration to estimate the location of the speaker SLOB relative to the listener SLOB. We calculate distance just as we saw before; normalize the audio samples, generate the same 300Hz sent signal discretized using the sampling rate of the listener SLOB, and compute the cross-correlation to obtain the most-likely index and resulting distance. Theory tells us we should receive a relative-localized coordinate point. This was not the case, however. We ran out of time trying to solve this road block, so we leave the solution to future work.

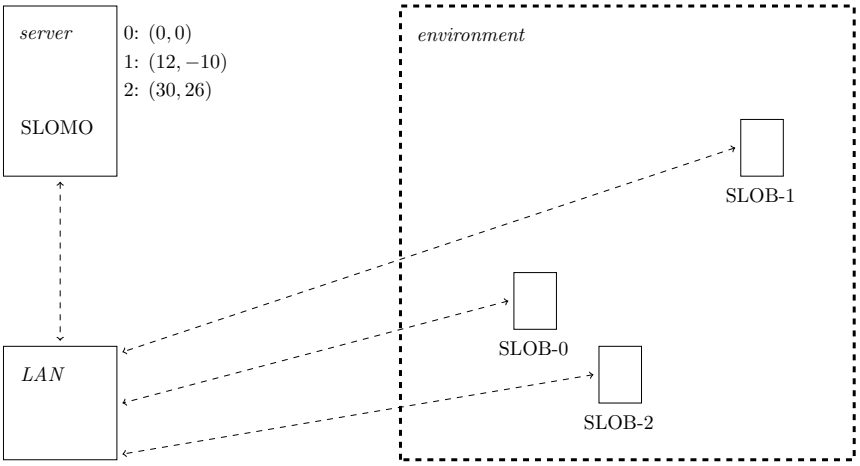


Figure 3: System architecture.

4. RESULTS

In this section, we detail some key observations and results from our proof-of-concept system. First, we observe our SLOBs achieved a maximum audio sampling rate around 1700-1800 samples per second per microphone. This is nearly identical to the theoretical maximum sampling rate for our architecture. Unfortunately, our system does not achieve the same theoretical distance fidelity. In fact, our distance calculations are lower-bounded in error by our time synchronization error. We could not realize our system with low enough time synchronization to multilaterate a point given a set of distances and base-stations. We provide figure 4 as example of one actual iteration of SLOMO.

Our general architecture is scalable and promising for future work. While we could not achieve our end goal in this paper, we hope to solve this in the future with better time synchronization.

5. LIMITATIONS

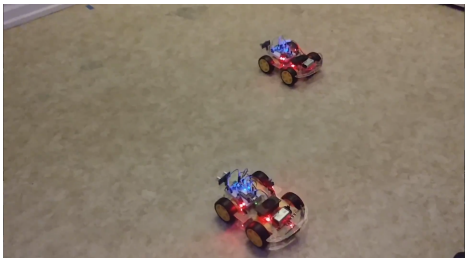
Not including the time synchronization roadblock, we ran into numerous limitations in this project. We discuss these limitations, and provide a bridge for future work, here.

5.1 Hardware

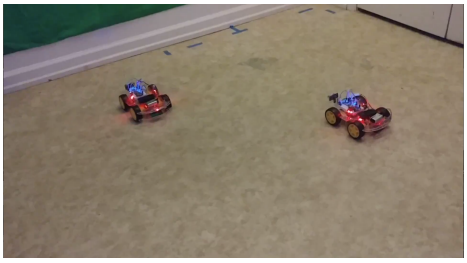
In our approach, we use two micro-controllers communicating over serial transmit and receive lines. We needed to do this because our Wi-Fi board only supported one analog pin, yet we had two microphones each requiring an analog pin. Furthermore, our MPU requires two analog pins for accelerometer and gyroscope data. Our Arduino board supports four analog pins, so we placed the four analog pins into the Arduino and factored our SLOB code to accommodate data transmissions between both boards. While at first this may appear to be a limitation, this is only true when using standard Arduino libraries. The serial library restricts baud rates to a maximum of 115200 bits per second. Yet, using custom low-level code, we could theoretically achieve 2 million baud. As discussed before, our baud rate already gives us tens of centimeters of distance accuracy, yet we leave higher baud (increased sampling rate) for future work. In an ideal world, we would custom-design a single micro-control board that handles all of our functionality.

We also ran into board-specific issues with the Wi-Fi board. The Wi-Fi board's internal OS implements a watchdog to periodically check its wireless connection. Because of this, the Wi-Fi board requires interrupts during code execution. The watchdog implements this by interrupting code execution during DELAYs. Furthermore, if the watchdog detects a loop that has been running for too long with no delays, then the board will restart itself. This potentially limited our audio sampling rate.

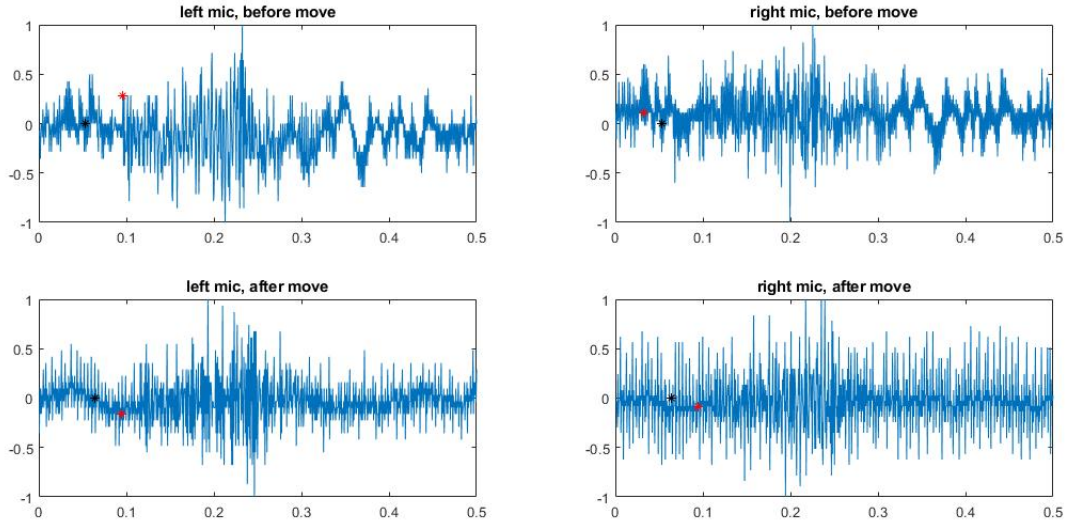
Finally, our SLOB architecture was severely limited by each board's memory capacity. The Arduino board contains 2KB of memory, which severely limits its internal buffer for microphone sampling. We solved this by directly streaming each byte of sampled audio directly to the Wi-Fi board over serial. This requires synchronization and specialized stream-handling code, which was challenging to implement. Fortunately, the Wi-Fi board contains 128KB of memory, which allowed us to save about 2048 left-right samples ($2 \cdot 2048 = 4096$ actual microphone samples, at two bytes each yields 8192 required bytes) – plenty of room for 500 milliseconds of audio sampling. Unfortunately, we still have to ship this data to the server. We send these samples via POST request, meaning we have to build a string in-memory to represent our data. Each sample corresponds to about 3 digits, which yields 3 characters per sample.



(a) Before moving forward 100 centimeters.



(b) After moving forward 100 centimeters.



(c) Actual localization microphone samples. In this run, the microphones were sampled for 500 milliseconds at 1734 samples per second. The red dots indicate the cross-correlation max-likelihood lag at which a 300Hz sine wave for 125 milliseconds appears. The black dots indicate the server-estimated time at which the speaker started broadcasting the 300Hz signal. The 300Hz estimated position is typically correct, yet the speaker start time is almost always incorrect. This is our most significant source of error, matching our theoretical insights.

Figure 4: Example execution of localizing one SLOB to another using SLOMO. The left SLOB is the leader/listener, and the right SLOB is the speaker.

Each character requires one byte of storage. Adding in comma separators, another byte, our POST data requires 8 times the memory capacity of the actual audio samples. This cuts our memory budget quite close. In future work, we might consider a custom board containing more memory, we might consider compressing our data before sending, and/or we might create a custom network transfer protocol to enable SLOB-server data streaming.

5.2 Approach

Sound is a useful technique for agent communication. Sound signals do not require line-of-sight, nor require a wireless connection. Sound signals can pass through a variety of materials, and is slow enough to theoretically judge precise distances. Despite the appeals of audio communication, there are a handful of drawbacks. First, sound may bounce around after hitting solid objects, which causes multipathing and reduces the arrival-time accuracy.

Finally, sound-to-distance estimation requires accurate time synchronization. Our approach is severely limited in this capacity. As we showed before, even 10 milliseconds of time-synchronization error yields above 3 meters of fidelity. GPS gives similar error, yet only works outside, which still gives this project technical merit since our technique works almost anywhere. Regardless, for centimeter-level accuracy, an entirely new time-synchronization method must be developed. We leave this to future work.

5.3 Project Completion

While we could not solve our intended research goal, we demonstrated significant progress and justified the merits of continuing. We could not complete our target goal due to time constraints – given another few months of testing, we certainly could have achieved real localization results.

6. REFERENCES

[1] https://github.com/jugoodma/818bw-project.

[2] Mani, S. K., Durairajan, R., Barford, P., and Sommers, J. A system for clock synchronization in an internet of things. *CoRR abs/1806.02474* (2018).