Local Multilateration via Time Difference of Arrivals in Multiple Audio Recordings

Gorbunov M. Michael, Mundkowsky L. Elizabeth (School: East Brunswick High School, East Brunswick, NJ, United States)

Introduction

We tackled the problem of accurate and affordable local positioning of targets (i.e. machinery, people walking, bouncing ball). Our goal was to use the time difference of arrival of sounds produced by a single sound across many microphones to determine the target's location in an 1x1 meter area with +-5 centimeter accuracy.

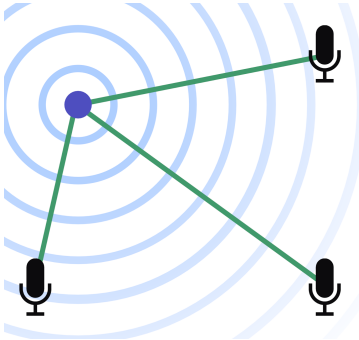
Some existing solutions for positioning include GPS, radar, lidar, odometry, SLAM. These are certainly really good solutions but they come with drawbacks. GPS and radar are accurate only at a large scale. Lidar and SLAM fail when visual obstructions are present, and can be expensive (namely lidar). Sound-based positioning, however, is less vulnerable to stationary obstructions and has been shown possible. Researchers have previously successfully located different species of birds in an open meadow by tracking the birds' songs. The researchers were also able to locate an artificial source sound composed of mixed real bird songs. Our goal is to recreate this result however, developing our own algorithm for they used paid tools for sound processing.

Methodology

For our solution, we chose to have many microphones and one sound source because it appeared more novel. Also, the sound source does not need to know of the data collection, and so could be useful for wildlife tracking or sonar.

We did find some research on related problems but ultimately none of it made it into the final demo. Our process was a matter of solving the next subproblem as we found them. For this proof of concept we had fairly ideal conditions. To develop further we would use more research, ex: for noise cancellation.

As shown right, our setup involved using many microphones and one sound source. Recording on all microphones at the same time, the sound will reach each microphone at slightly different times. This is because the sound source is at a different distance from each microphone. The Difference In Distances (DIDs) between different microphone pairs is used to reconstruct the position of the sound.



Algorithm Explanation

Inputs

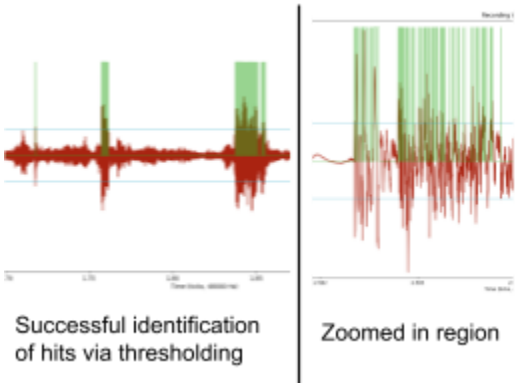
This algorithm works with multiple audio recordings of the same sound, i.e. the same events captured from multiple microphones. Before being processed, some manual work is required to align the start-time of recordings to be within 0.1 seconds of each other. Also, each audio sample needs a corresponding noise profile.

For positioning regression to work, the positions of the microphones need to be known, as well as the speed of sound. In our case, we used 343.8 m/s which corresponds with 70 degree air.

Hit Detection

The audio recordings for which this algorithm is designed consist of ‘hit’ sounds at different intervals, generated by hitting a lego and battery together. Extracting the timing of these hits is done purely in the amplitude domain.

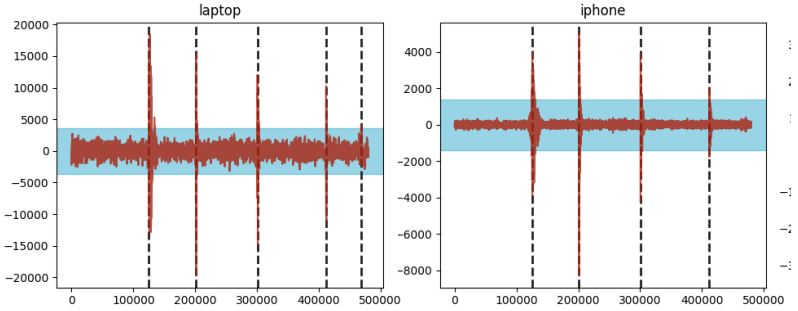
First, the noise profile is used to generate a threshold. Whenever the waveform exceeds this threshold, it is louder than the background noise and registered as part of a hit. The threshold is simply the maximum of the noise profile x 1.1.



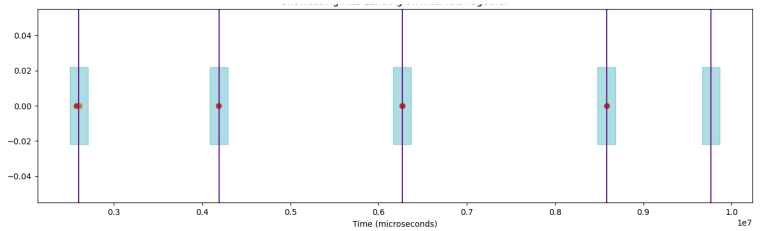
Next, every point in the original recording that exceeds the threshold is marked. Actually zooming into a green region, the result is noisy because the waveform oscillates. To clean this up, a sustain filter is applied which only records a new hit if there has been sufficient time since the threshold was previously exceeded.

Hit Grouping

Extracting the times of hits from multiple recordings, the next problem is how to group them. They each happen at slightly different times, and sometimes background noise will register a hit on one recording but not another.



To group them, one track is arbitrarily chosen as the master track, and other tracks are compared to it. For each master hit, an interval of acceptable timings is created on which to search for other hit times. If it finds that every track has a hit within the interval, those hit times are grouped.



The fifth interval has no dots in it. This agrees with the waveform, wherein the final laptop hit is not found in the iPhone recording.

Difference In Distance (DID) Calculations

Grouping the hit-times produces a time difference between hits for each audio recording. Time differences between two recordings is the result of four main factors:

$$\text{Time Difference} = \text{TDOA} + \text{TDOA Drift} + \text{Alignment Issues} + \text{Variance}$$

In the previous step, Time Differences were found. This step we extract Time Difference of Arrivals (TDOAs), i.e. differences due solely to the location of the sound. We control for drift and alignment. Variance is mitigated by taking many samples, but cannot be removed.

TDOA drift is a phenomenon we discovered while processing the data. Likely due to small precision errors, two recordings - even at the same sample rate - will misalign over time. The largest drift in our microphone pairings is 1.6×10^{-5} seconds of misalignment per second. Quantifying drift was done by looking at the time difference at the same location over time, which without drift would be a flat line. Adjusting for TDOA drift is simple. The drift value explains how much two recordings shift every second. Because each hit is time-stamped, we can determine how much drift has misaligned two recordings.

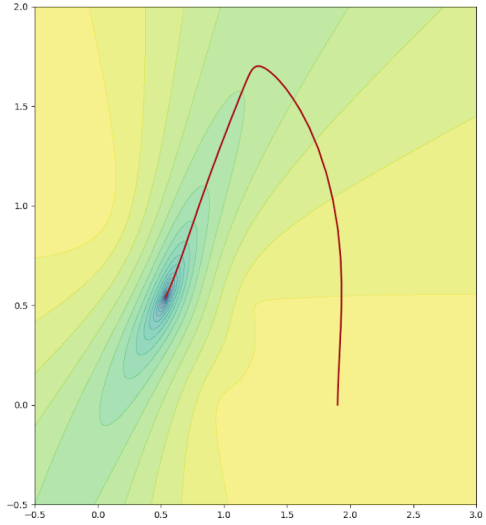
During pre-processing, the recordings are manually lined up and thus have some offset solely due to poor alignment. Honing in the alignment to acceptable levels is done by using time differences of a known point. For example, in the test data presented, ten hits were made at a point equidistant to all microphones i.e. those points have an expected offset of 0 seconds. For one microphone pair, those time differences averaged 17 µS, so 17 µS was subtracted from all time differences for that pair.

Converting TDOA to a Difference In Distance (DID) is done by multiplying by the speed of sound. If two microphones have a TDOA of 1 second, that sound is 343 m closer to one microphone than the other.

Position Regression

The final step is to take the DIDs (extracted above) and find the original point. This is done with gradient descent over an error function.

The error function chosen is fairly simple. At every x & y position in the plane, it is easy to calculate the distance to each microphone. Then, a theoretical DID, i.e. the DID from this specific x & y for every microphone pair can be calculated. Now, we have the DID that would be produced at this coordinate, as well as the empirical DIDs. Summing the square differences for each microphone-pair is our error function. Then gradient descent can be trivially performed.

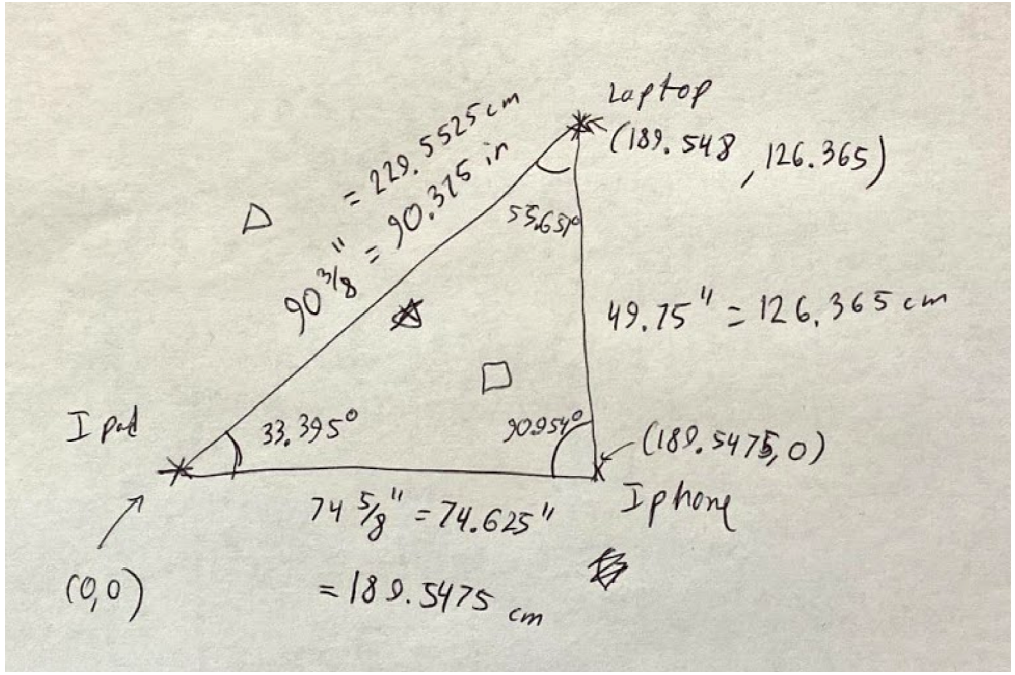


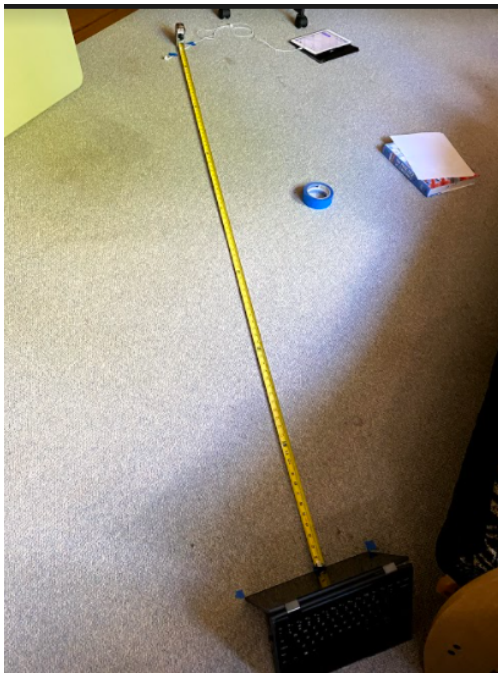
$$Error(x,y) = \sum^{mic-pair} (DID_{empirical} - DID_{from-coordinate})^2$$

Results

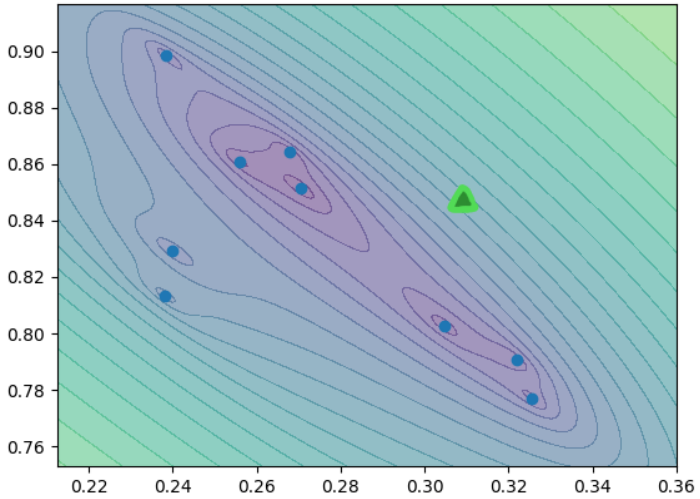
Position Predictions

To test our algorithm, we generated test data by creating sounds in known positions. Microphone positions were determined via triangulation with the side lengths. Two of the test positions were similarly computed. One position however was done in reverse - it was specifically chosen to be equidistant to all microphones, and is used for alignment calibration. Below is a diagram of the microphone placements, as well as sketches of the three locations tested, marked with a triangle, star, and square symbol.

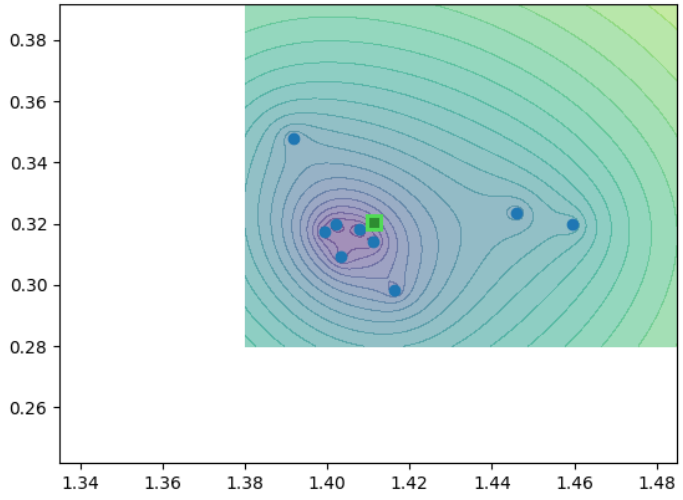




Our algorithm produced the following predictions:



Triangle location



Square location

The neon green symbols indicate the position predicted via measurements and triangulation. Both figures have roughly the same scale - triangle positions have more spread. Note, the predictions for the star location are not shown. Those positions were used for calibration so they are correct by definition.

Below are quantitative results for the average of these two clumps.

	Avg Predicted X	Avg Predicted Y	Measured X	Measured Y	Distance (cm)
Triangle	0.274	0.832	0.3083	0.846	3.734
Square	1.415	0.319	1.41268	0.3206	0.318

The margins of error for the measured values are greater than the number of decimals suggests, being around +- .02 m.

Discussion

Significance

Our results show that affordable, sufficiently accurate localization by sound from any source can be achieved and suggests that localization in real time is possible. We do not have data to determine the true error of our approach but our data does agree fairly well with triangulation-based measurements. Nevertheless, our prototype still suggests that acoustic localization can be accurate on a small scale.

We already knew that local positioning from TDOAs is feasible from an existing study that used bird calls to identify birds, but our results expand on this by extending to any percussive sound. Any sound louder than a certain threshold above background noise can be identified and have its TDOA extracted, as we did with the battery striking the lego. The bird call positioning study also used the Canary sound analysis package, which costs $200, to extract the TDOAs. Our prototype improves on the cost of acoustic localization because we wrote our algorithm to identify the attacks and extract the TDOAs at no cost. However, because our algorithm is so simple it would fail when hit sounds are made too close to each other, and would need some tuning when working on larger distances, due to the larger time differences.

Issues

Initially, we planned to extract the TDOAs using the frequency domain of the recordings, but this proved problematic when we found some of the background noise shared frequencies with our target sound. After revising our hit detection algorithm to rely on the amplitude domain instead, however, this issue was resolved and another issue arose. Using the amplitude domain, we needed to set different thresholds for each recording because the background noise was not at the same level for all the

recordings. However, we solved this by looking at the noise profiles of the recordings to determine the thresholds as explained in hit detection.

One unexpected issue we encountered was the difference in sampling rate between the iPad and iPhone because the difference in tick frequency made it impossible to perfectly align those recordings. We resolved this by converting the time units of all the recordings to microseconds, with 1 tick equaling 20.833 uS at 48kHz and a tick equaling 22.675 uS at 44.1kHz. We only did this conversion once because it introduces rounding error as the microseconds are stored as integers.

The drift in the TDOAs was also unexpected. We expected small random changes or no changes in the TDOAs because the source sound was stationary. With our calibration set up specifically, we expected there to be small to zero difference between the TDOAs because the microphones were equidistant from the battery and lego. However, interestingly, the difference in TDOAs generally increased as time passed even though the source sound was stationary. To account for this, we found how much the TDOAs misaligned from hit to hit, the drift value, and subtracted this drift value from the TDOAs.

We believe our prototype could be expanded to function in real-time. No time is required in generating a frequency domain because we rely on the amplitude domain, and noise reduction is not necessary either as long as the target sound is adequately loud. One obstacle in functioning in real time, however, is that the audio data being recorded by the microphones must be transferred to a computer quickly. There would also be some necessary delay before the source sound’s position could be predicted because enough time needs to pass to allow for the TDOA drift to happen so we can calculate drift value.

Conclusions

Our proof of concept demonstrates successful localization of sound. Thresholding for hit detection is certainly somewhat crude but it works with our somewhat ideal audio. We were hoping to be able to do more work with signal processing, potentially introducing noise or doing pattern recognition to cross reference among recordings. However, all other parts of the algorithm are as sophisticated as they need to be, and our variance falls within our goal.

Potential Applications

A similar solution has already been used for wildlife tracking, specifically locating birds from their calls.

If water-proofed, our work could also be used in marine environments or generally obscured areas like forests where sound travels better and the targets are not visible. With sonar specifically, a fleet of submarines each acting as a microphone could listen to incoming sound waves (from wildlife or other submarine’s sonar pings) and use our algorithm to determine where the original sound was coming from.

References

While many of these items see no apparent influence on the final result, they were useful tangents while we considered which specific approach to take.

Accuracy of a Passive Acoustic Location System (Mcgregor, et al) - Bird positioning via similar technique

These three links would be heavily used if development continued to work with noisier recordings
Adaptive Whitening for Improved Real-Time Audio Onset Detection (Stowell, Plumbley)
Audacity noise removal algorithm
Onset detection algorithm specifically for music

Sonic Visualizer - was useful to see our options and gut check that a certain approach could work
Matplotlib - almost all graphs were generated via Python