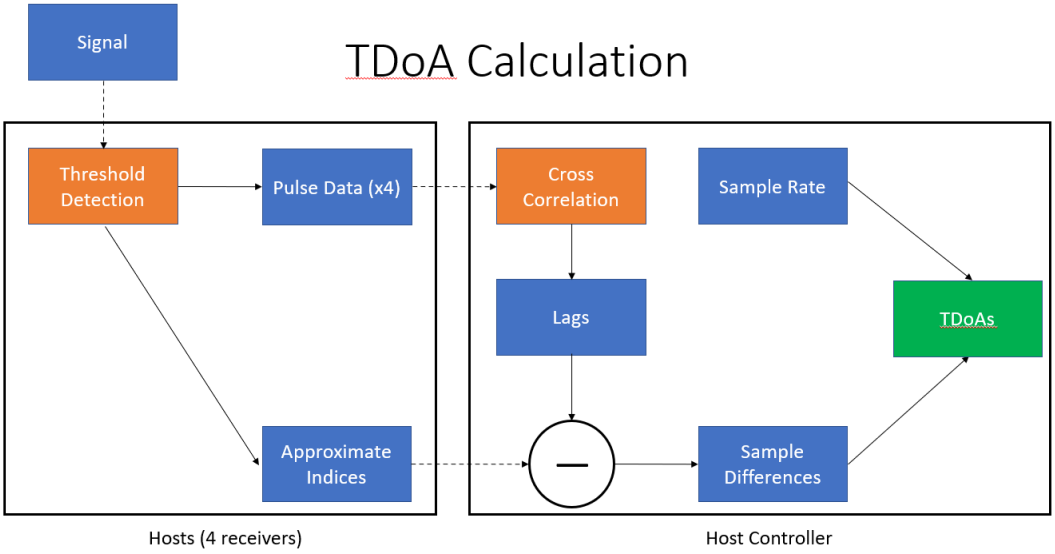
6/8/2023

Localization Documentation

WES 207 (Spring 23) – Lizard Localization

Ryan A Margraf

Overview

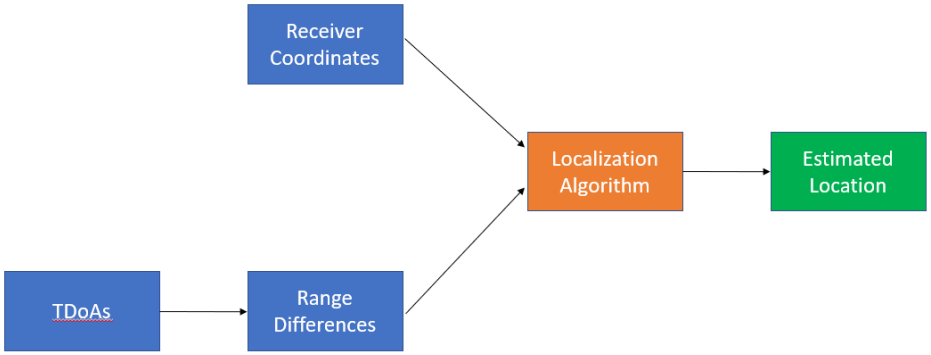


To calculate the position estimate from the data provided by each host, we used an algorithm that uses Time Differences of Arrival, or TDoAs, in which we use the time delay observed between when the pulse arrives at each receiver to determine the distance to the transmitter.

The threshold detection provides approximate times of arrival, but we need them to be as accurate as possible. To achieve this, the pulse data from each receiver, along with its respective estimated index, is sent to the host controller. The host controller performs a cross correlation between the pulses, finds the peak index, or lag, then takes the difference between that value and the estimated index from the threshold detection. That difference is then divided by the sample rate to get the TDoAs. We decided that performing the cross correlation on the actual pulses received instead of against a template signal would be better because the signal was not reliably the same, and so a better estimate could be made this way.

This process corrects most of the timing difference between the pulses, so the inputs to the localization algorithm are more accurate.

Localization



For localization, the calculated Time Differences of Arrival are multiplied by the speed of light to obtain the range differences between each pair of receivers, that is, how much further the pulse is estimated to have traveled to one receiver over another. Using this information along with the receiver locations, the location of the transmitter, and thus the lizard, can be estimated.

The algorithm used is the one described in "Passive Source Localization Employing Intersecting Spherical Surfaces from Time-of-Arrival Differences" by Schau & Robinson. “The Spherical Interpolation Method of Source Localization” by Smith & Abel was also referenced in creating the code, and so some variable names come from that.

Simulations

The following MATLAB scripts are included in this github repository:

LocTest2D.m

This is for testing the localization algorithm. It generates a random location, adds noise, estimates the TDoAs, then runs the algorithm. It outputs a plot showing the location of the four receivers (blue stars), the hyperbolas showing the range differences for each pair of receivers, the location of the transmitter (red x), and the location estimate (black circle). Additionally, it outputs the magnitude of the x and y dimension localization error. The receiver locations are adjustable using the parameters at the top of the script, and the noise level of the range estimates can also be adjusted.

LocErrorTest.m

This script is for testing the error of the localization algorithm. Given the specified receiver locations, heights, and noise variance, it runs a specified number of trials and finds the mean and median error of the localization estimate. It also checks the accuracy of using the “plus” solution to the quadratic equation in the specific algorithm we used.

ccorr.m

This script implements the cross correlation as it would be implemented in C++.

ccorrTest.m

A simple test script for the above. Compare outputs with the xcorrTest C++ app for testing.

LocTest2Receivers.m

This is useful for comparing 2 captures to make sure they are relatively synced. If they are, the location estimate should appear close to the middle of the output plot (50,50).

Compare_capture_ccorr.m

This script performs the TDoA calculation between 2 data files via cross correlation. It checks both filtered and unfiltered data. It also shows some plots of the signals.

C++ Code

TDoA Calculation

The first step to the TDoA calculation is the cross correlation between each pulse received by the host controller. The Naïve method was found to be too slow, so instead of convolution, the FFT of each pulse is taken, one of those results is conjugated, point by point multiplication is performed, and then the IFFT is taken on the result of that. The xcorrTest app provides testing for the cross correlation and the outputs can be compared to the MATLAB function outputs using ccorrTest.m. Note that since the FFT library used, FFTW, does not normalize the outputs, they will be off from the MATLAB simulation by a factor of the FFT size.

After cross correlation, the maximum lag is subtracted from the sample difference in the threshold estimate to get the number of samples that separate the arrival of each signal. These are then divided by the sample rate to obtain the TDoAs.

Localization

The C++ code implements the localization algorithm quite similarly to the MATLAB simulation. It uses the Eigen library. A test app, locTest, is provided to confirm that the script matches the MATLAB output from LocTest2D.

Findings

Localization Simulation

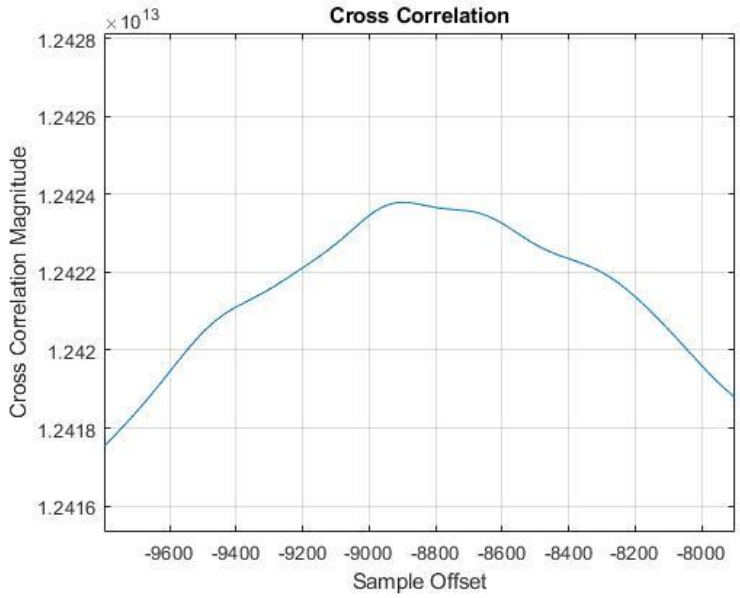
Max Height (m)	10	10	20	20	20	30	30	30	50	50
Delta (m)	2	3	4	5	6	7	8	9	12	15
Min Height (m)	4	1	8	5	2	9	6	3	14	5
Mean Error (m)	17.2085	9.7016	7.4875	6.662	5.1129	5.3368	4.9419	4.5924	5.1984	4.5027
Median Error (m)	6.3454	4.8475	4.2695	3.9551	3.7478	3.8864	3.7547	3.6364	4.0504	3.7241
Est1>Est2	98.05%	99.21%	99.30%	99.64%	99.83%	99.82%	99.87%	99.90%	99.90%	99.90%

The above chart shows the error measurements and 1 solution reliability of the quadratic equation for 100,000 trials and 3m standard deviation of noise for range estimates obtained by running the LocErrorTest.m script. It can be seen that the “plus” solution of the quadratic is more accurate the

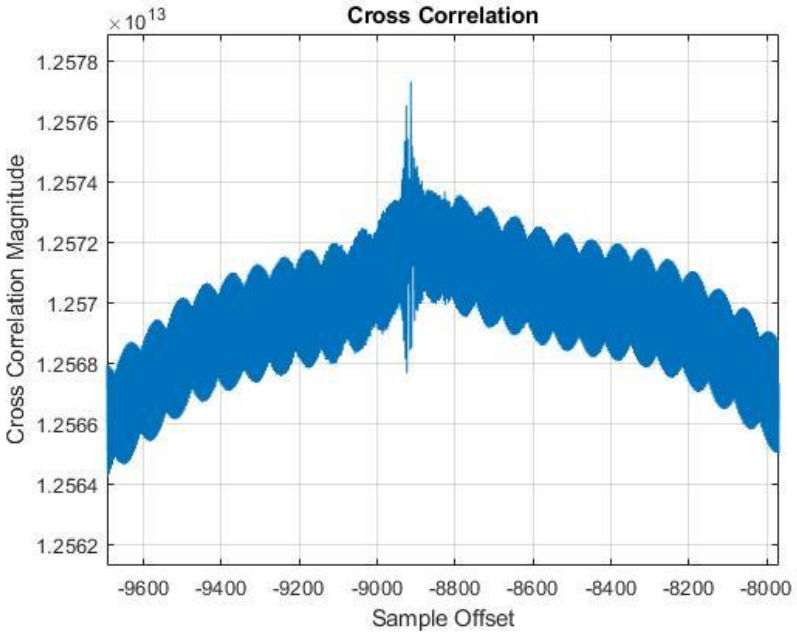
vast majority of the time. I examined some of the situations where it was not better, and it is very close, so it should be fine to just use the “plus” solution.

Another important observation from these results is that the error decreases with a greater height spread between the receivers, as well as the closer that one of them is to the ground.

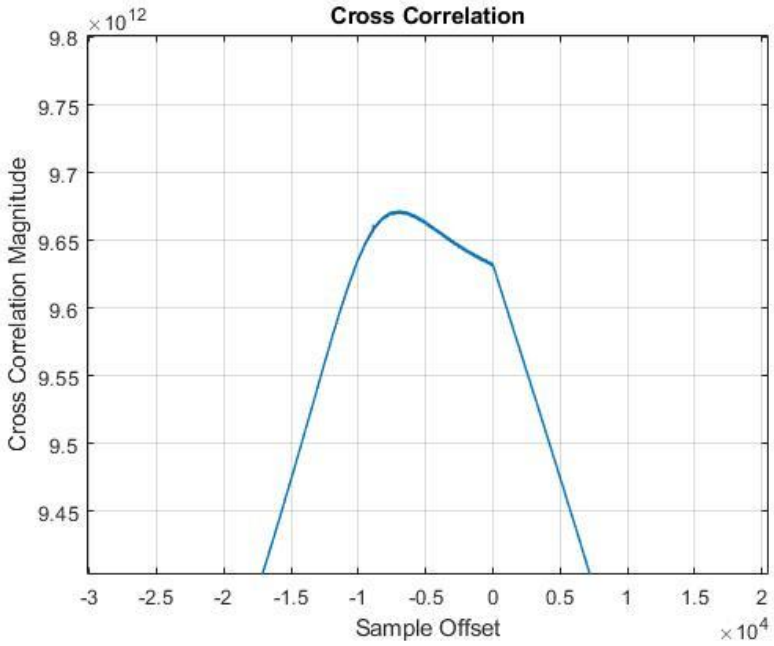
TDoA Calculation



On a test with two receivers right next to each other and the transmitter less than a meter away, the cross correlation on the filtered pulses provided a sample offset of 15, or a TDoA of about 600ns. Due to the weak correlation properties of a sinusoidal signal and the smoothing of the filter, it is not surprising that this is slightly off.



Performing the cross correlation on the unfiltered pulses instead results in a noisier plot, but a spike at the peak. This corrects the difference to only one sample, which shows that the system is indeed synced. A theory we have is that due to the pulse not being filtered, the ramp up time near the beginning and end is reduced and makes point of maximum overlap more sharp. However, it is also possible that this could be from modulated interfering signals as discussed later. We suggest testing in an environment free from this type of interference if possible.

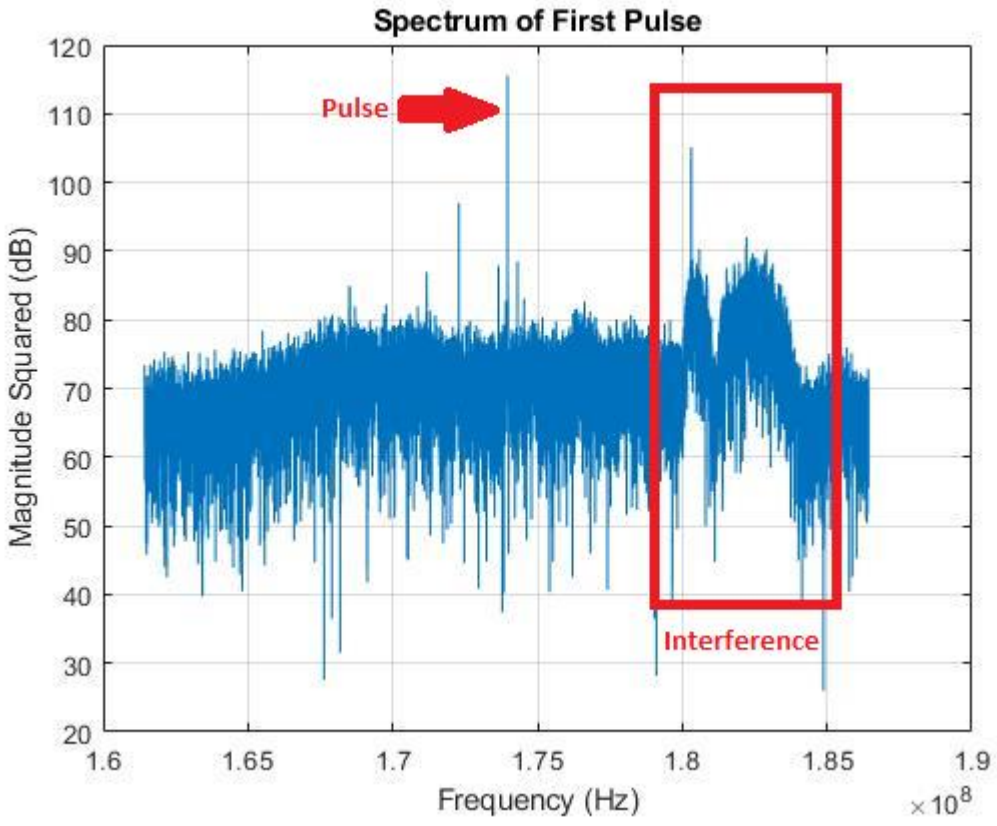


We also tried using a smaller part of the pulse for cross correlation, but it performed significantly worse. For example, using 20ms of data instead of 30ms increased the TDoA error to tens of microseconds, resulting in a range error of several kilometers. We thus recommend sending the whole pulse to the host controller for cross correlation, plus some time on either side, unless a signal with better autocorrelation properties could be used.

Issues

Interference

The system requires a high sampling rate to obtain adequate range resolution. This not only reduces the SNR by letting in more thermal noise due to the high bandwidth, but also increases the susceptibility to interference. The below figure shows interference in a captured pulse on UCSD campus.



The center frequency of the transmitter we tested was very close to VHF TV channels 7+, which begin at 174 MHz[1]. It is unknown if this issue will be present at the desired deployment location of the system. In our testing, channel 8 seems to be the dominant frequency, and the interference

[1] https://www.rfwireless-world.com/Tutorials/TV-channel-frequencies.html

can be mitigated by filtering, though this may not match the situation at the deployment location. If channel 7 happens to be in use, filtering that out may be less feasible due to how close it is in frequency to the transmitter.

Timing

The GPSDOs have a timing noise standard deviation of 50ns, which corresponds to about 15m, so getting better range resolution than that requires improvements to some aspect of the system. One suggestion would be to average multiple localization estimates over a moderate period of time, perhaps a few minutes.