Assignment 6: Bearing Angle Estimation

Paul Monroy
University of California, Santa Barbara
Santa Barbara, CA 93106
paulmonroy@ucsb.edu

Abstract

Through trigonometry, we are able to figure out the delay of certain signals. We implement this by analyzing either a delay in two audio signals that were transmitted, as well as making a delay in audio.

A. Introduction

During a signaling cycle, the transmitter sends out a complete sequence of N coherent signals, stepping through a defined frequency band with frequency increment Δf. Through trigonometry, we are able to figure out the delay of certain signals. We implement this by analyzing either a delay in two audio signals that were transmitted. This delay is very small, but still apparent enough in sound signals. Through our knowledge of bearing angle and correlation, we find the delay in audio signals and the transmitted angle. In the following problem, we take a mono sound signal and transmit in stereo form. We do this by implementing a delay in the audio.

Part A: Bearing-Angle estimation

1. Problem 1: Performing cross-correlation of the two data tracks and plot the resultant correlation function

In this problem, we are to consider a passive detection system consisting of two acoustic receivers. The separation between the receivers is 2.5 meters. The acoustic propagation speed in air is 343.6 m/s. The acoustic data tracks detected by the receivers are given. The sampling rate of the A/D conversion is 48 kHz.

We are to perform cross-correlation of the two data tracks shown in **Figure 1**.

Cross correlation of two functions f(t) and g(t) is defined as

$$R_{fg}(t) = \int_{-\infty}^{\infty} f(\tau)\, \mathrm{g}^*(\tau - t)\, d\tau$$

It could also be said the cross-correlation function can also be stated as a convolution of f(t) and the conjugate of g(t).

$$R_{fg}(t) = f(t) * g^*(-t)$$

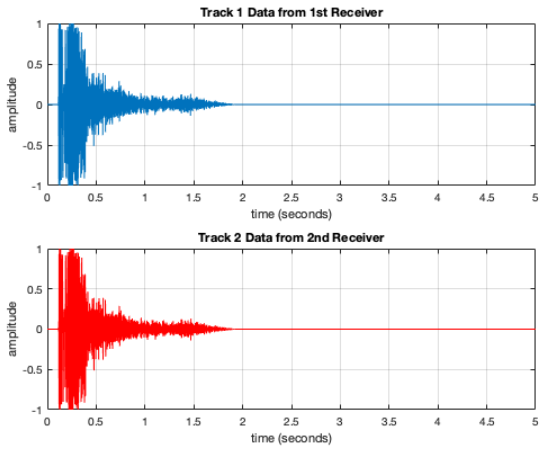

Figure 1

Using the *xcorr* function in MATLAB, it returns two variables: the cross-correlation result *r* and *lags*, where the delays through which each convolution iteration takes place. The lags usually take place between -length(r) and +length(r). Finally, the time delay is just the lags evenly divided by the sampling frequency.

```
[r, lags] = xcorr(x1,x2); %%% compute cross correlation %%%
t_shift = lags/Fs; %%% compute time vector for positive and negative delay values %%%
```

The cross-correlation plot is shown in **Figure 2**. We see as we get close to zero time delay, the two signals become a lot more similar.

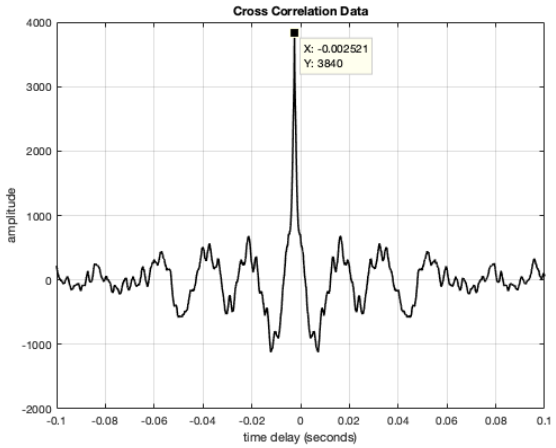

Figure 2

2. Observing the accuracy of the time-delay estimation and its relationship to the bandwidth of the power spectrum

In the final section of the code, we look for the maximum value as well as the maximum index of *r*, defined by `max_value` and `max_index`, respectively. We then find where the overall time delay of the two signals is by matching the maximum value *r* index with the corresponding time-domain index, defined as `t_max`. The value of this time delay is -0.0025. This means that track 2 is delayed by 0.0025 seconds. This corresponds to the spectrum because it is at the peak value at delay = -0.0025s

```
[max_value, max_index] = max(abs(r));    % find maximum of cross-correlation
t_max = t_shift(max_index); % time delay
theta = asin(343.6*t_max/2.5); %%% compute bearing angle in degrees here %%%
theta_deg = theta*180/pi;

theta_spectrum = asin(343.6*t_shift/2.5);
figure
plot(theta_spectrum, r)
xlabel('Bearing Angle (radians)')
ylabel('Cross Correlation Amplitude')
title('Cross Correlation as a function of angle')
```

3. Plotting the bearing-angle estimation profile of the acoustic source by rescaling the time-delay profile.

In the same block of code, I write an expression defining bearing angle. Referring to **Figure 3**, if the separation between two rays is D, then the distance offset between the two is

$$\Delta x = D sin\theta$$

Figure 3

We define the time delay τ as the distance delay divided by the speed of the medium where the rays travel.

$$\tau = \Delta x / v = D sin\theta$$

Rearranging, we find the bearing angle θ.

$$\theta = \sin^{-1}\left(\frac{v\tau}{D}\right)$$

This is implemented in the MATLAB code, with v = 343.6 m/s and D = 2.5m. The bearing angle is labeled `theta` and its value is -0.3538 radians or -20.2711°.

The cross-correlation spectrum is plotted against angle in **Figure 4**. The peak of the plot is located at x-value of -0.3538 radians, where our calculated bearing angle is.

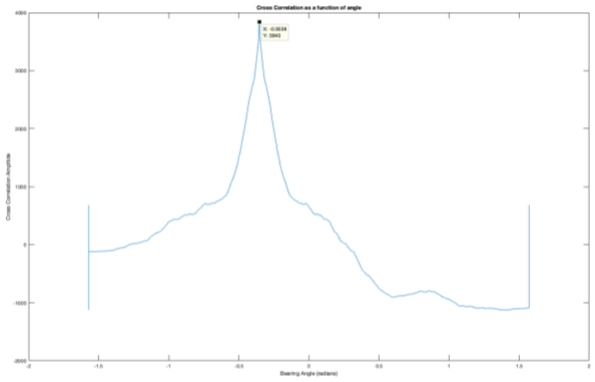

Figure 4

Part B: Rolling Thunder

We are given a single-channel (mono) soundtrack. The objective is to produce a stereo soundtrack such that the acoustic source seems to move steadily from -60° to +60°.

4. Describe the concept of your approach.
5. Implementing the technique.

Here, I combine problems 4 and 5 where I explain my understanding of TA Connor Sanchez's starter code. I

In this problem, we start off by reading the Thunderstorm_6s.wav file. We set ear spacing = 0.18m, speed of sound = 343.6 m/s and our degree to 60°. The audio track will move from +60° to -60°.

```
speed_sound = 343.6;  % Speed of sound:
343.6 m/s
ear_spacing = 0.18;  % Spacing between
ears: 18 cm
deg = 60;  % Source will rotate from -deg
to +deg.
rad = deg2rad(deg);  % Convert from degrees
to radians.
```

We then find a way to linearly change the source angle by scaling the angle `rad`, in radians, by 2/[length(x)-1] and shifting it by `-rad`.

```
[len_x, ~] = size(x);

a = 2 * rad / (len_x - 1);

b = -rad;
```

Next, we take the indices of x and apply our linear shift: `an+b`. Finding the max index delay possible with *`max delay`*, we are able to scale and attain a *`delay`* vector by scaling `sin(`*`theta`*`)` by *`max_delay`*.

```
n = (1:len_x).';  % 'Sample indices.
theta = a * n + b;  % Linearly changing
source angle.
max_delay = fs * ear_spacing / speed_sound;
% Maximum delay possible.
delay = max_delay * sin(theta);  % Delay
vector.
```

We then find the left mapping by going from *`n`* to *`n - delay`*. Rounding performs nearest-neighbor interpolation.

```
n_left = round(n - delay);
```

We then prune out the out of range frequencies by finding the indices where `n_left` is greater than or equal to 1 and less than or equal to *`len_x`*. This is shown in the variable *`good_left`*. We then make `n_left` into these indices. We set a variable *`n1`* into the values of *`n`* using the indices *`good_left`*.

```
good_left = find((1 <= n_left) & (n_left <=
len_x));

n_left = n_left(good_left);

n1 = n(good_left);
```

We then insert zeros for delays to initiate our vector *x1* which will be our left ear audio.

```
x1 = zeros(len_x, 1);
```

Finally, we map the indices. We advance the left signal by putting the good indices into `x1` as the delay left ear indices of `x.` This is where I filled in the code.

```
x1(n1) =  x(n_left);  % TODO(students):
Delay (or advance) only the left signal.
x2 = x;  % TODO(students): Set the right
signal to the original.
```

We have achieved our delayed audio. I save it and test it with *sound* to verify that it is correct.

```
audiowrite('RollingThunder.wav', [x1,x2],
fs);

[y, fs] = audioread('RollingThunder.wav');
sound(y,fs)
```

After hearing the replayed audio, I could indeed hear a delay and the audio traveling from left ear to the right ear.

B. Summary

Here we covered the topic of bearing angle and correlation. More specifically, we implemented cross correlation as well. In the first problem, we analyze a delay in two audio signals and attempt to find the delay using cross correlation. We then find the bearing angle. In the second problem, we delay a mono track to make it stereo. By doing this, we are able to hear a sound that rotates from one ear to the other.

Through analysis of these problems, it is much clearer to see how correlation is useful, especially when finding a delay between two or more signals. It is also helpful to see how mono audio can be turned into stereo audio.