EE16A Lab: Acoustic Positioning System 1

TA, ASE, ASE, ASE
Announcements

- All software; no hardware involved
  - That means you can finish at home!
  - Just APS 1 though, come in for APS2
Where Are We Now?

- Imaging Module
- Touchscreen Module
- APS Module
Today’s Lab: Acoustic Positioning System

- Global Positioning System (GPS)
  - Basically the same thing
  - Uses radio waves instead of sound waves
- Understand mathematical tools used for sifting and detecting signals
  - Think about cross correlation!
GPS?

- Satellites send signals at known times (beacons are synchronized)
  - But we aren’t synchronized to the beacons
- Receiver (us) gets these signals
- From **time-delay** of a beacon signal, receiver calculates **distance** to the beacon
- From **distances** to satellites, **position** is determined by **lateration**
- **How many beacons do you need to determine your location in 2D?**
Broadcast beacons → Received signal → Cross Correlation with Each Beacon

Calculate time delays of arrival (TDOAs) and position → Separate and average demodulated signals
**Time of Flight**

- Receiver gets signals from multiple satellites at the same time
  - Each is a known beacon/waveform
  - Periodic
  - We also know where the satellites are
- The receiver then determines when each beacon is received, with reference to when other beacons are received
  - Harder than it seems! Why?
Problem

- Our antenna receives all the signals at once
  - We have to separate out the useful information
- We have no clue when the satellites sent their signals
  - Signals repeat every 230ms
  - Because of this we can’t use the start of the recording as a reference
- Even if we can separate the info, we can’t just wait until we receive something, because we don’t know when it was sent
  - This week we will cheat a little bit, next time we’ll see how to really handle it
Recall: Inner (Dot) Product

- A mathematical operation for vectors
- One way to think about it is that it computes how similar two vectors are

\[
\langle \mathbf{x}, \mathbf{y} \rangle \equiv \mathbf{x} \cdot \mathbf{y} \equiv \mathbf{x}^T \mathbf{y}
\]

\[
= \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
= x_1 y_1 + x_2 y_2 + \cdots + x_n y_n
= \sum_{i=1}^{n} x_i y_i
\]
Recall: Inner (Dot) Product

- A mathematical operation for vectors
- One way to think about it is that it computes how similar two vectors are

Given this expression, and assuming $||x|| = ||y|| = 1$, when is this expression maximum?

$$\langle x, y \rangle = ||x|| \cdot ||y|| \cdot \cos \theta$$

An alternate form of the dot product

The value is maximized when $\theta = 0$. This is when the vectors point in the **SAME DIRECTION**, which is to say, the vectors are the **SAME SIGNAL**.

Thus the bigger the dot product, the more “similar” the two vectors are.
Tool: Cross-correlation

\[ \text{corr}(f, g)[n] = \sum_k f[k]g[k - n] \]

- Mathematical tool for finding similarities between signals
- Idea: Take \( g \) and slide over \( f \), compute dot product, slide again
  - Gets plotted with the shift amount
- From the previous slide, peak of cross-correlation tells us which shift amount makes \( g \) “most similar” to \( f \)
Tool: Cross-correlation

- “Sliding Dot Product”?
- Helps us find a specific signal midst a mix of many signals
  - Dot product computes similarity
  - Sliding dot product tells us how similar two signals are for a given shift amount (see gif)
- Use it to decode ambiguous texts from your crush
- At how many offset samples is the signal most similar?
How Will We Use It?

● Cross correlating should tell us where our beacons arrived in our signal
● From there we can try to find a way to compute the time delays
  ○ Then we can find the distances!
Solution Attempt

- Let’s cross-correlate each of the known beacon signals with what we recorded and plot the result
  - What do you expect to see?
Ok, What Now?

- Great! We can clearly see where each signal is in our received waveform
- Unfortunately we’re still not quite there… This doesn’t tell us much
- Idea: we don’t know when the beacons arrived, but based off of the offsets we know how much longer it took for beacon 1 to arrive RELATIVE to beacon 0!
- Let’s shift our axis so beacon 0 is at 0
  - We could pick any beacon to be the center. 0 is arbitrary
First we separate the signals

Then we shift
Now beacon 0 is our “origin” and all computations can be done relative to the new “0”
Relative offset of beacon 1

Shifted Cross-correlated outputs centered about Beacon0
Shifted Beacons

- We know the rate at which we recorded samples, and we know how many samples each beacon is from beacon 0.
- Since sampling frequency is samples/second, then
  \[
  \frac{\text{samples}}{f_s} = \frac{\text{samples}}{\text{second}} = \text{seconds}
  \]
- We know how long relative to beacon 0 it took for every other beacon to arrive.
- We know where the satellites are, so we can use the distances to find our location!

- Or can we..?
And Finally, Computing Distances??

- distance = rate x time
  - For beacons 1 through N, we know the time it took to travel
  - We know how fast various types of waves travel in air (AKA rate)
  - We can directly compute distance!
    - RELATIVE to beacon 0, not what we want
    - Oh, I guess we haven’t quite solved it yet
Actually wait, one more problem

- We know how long it took for beacon 1 to arrive AFTER beacon 0.
- If we magically knew beacon 0 arrived 4s into our recording, and beacon 1 arrived 3s after that, how long did it take for beacon 1 to arrive?
  - Knowing the time beacon 0 arrived ($t_0$) we can fully compute our distance
- But in general, we don’t know when beacon 0 arrived. You’ll be given it for today.
Notes + Next Lab:

- If we knew distance / time of flight for beacon 0, finding location is easy
  - Today this value will be given to you for testing purposes
  - Find out how to deal with this in APS2!

- It’s a long lab, don’t feel pressured to finish it here.
  - Can go home and get help from friends/TAs

- Note: Sliders in the notebook should but may not work; not essential so you can move on