EECS16A
Acoustic Positioning System 1

TA, ASE, ASE, ASE

Where Are We Now?

~~Imaging Module~~

~~Touchscreen Module~~

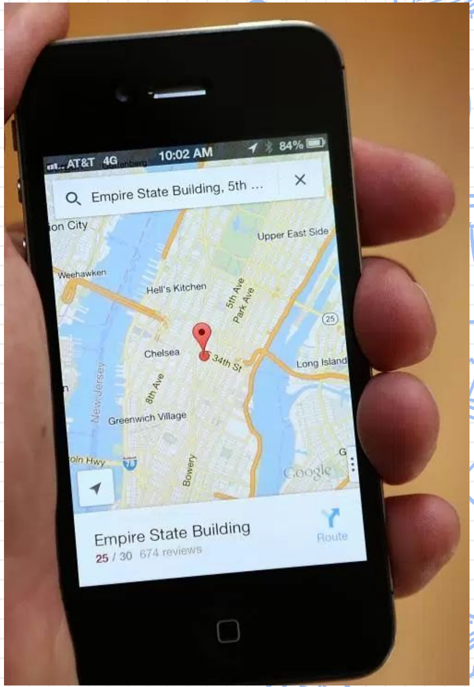
APS Module

Announcements

- All software
- Submit PDF of completed lab + checkoff quiz on Gradescope
  - due on Friday 4/24 at 11:59 PM PT

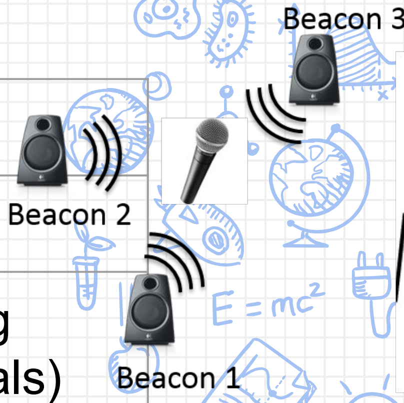
Today's lab: Acoustic positioning system

- Global Positioning System (GPS)
  - Very similar logic
  - Uses radio waves instead of sound waves
- Understand mathematical tools used for shifting and detecting signals
  - Think about cross correlation!



Set-up

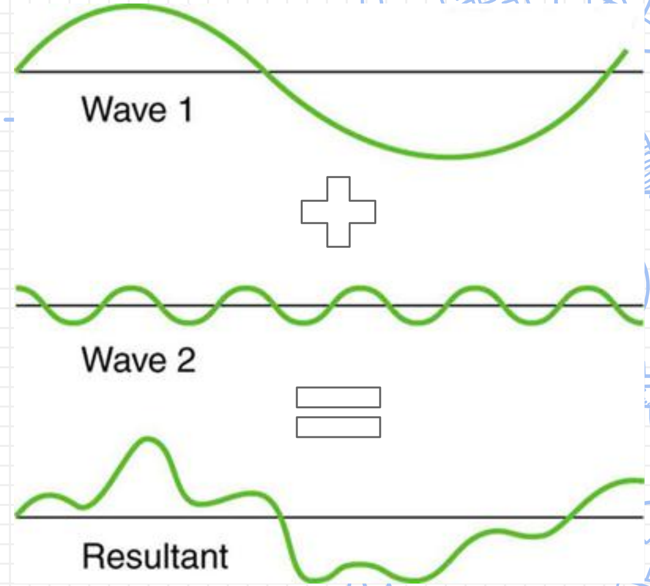
General	Lab Specific
receiver	microphone
Satellites repeatedly transmitting specific beacon signals	Speakers repeatedly playing specific tones (beacon signals)



- Known: Location of each satellite and what beacon signal each satellite is playing
- Unknown: Location of receiver ← what we want to figure out!

Set-up

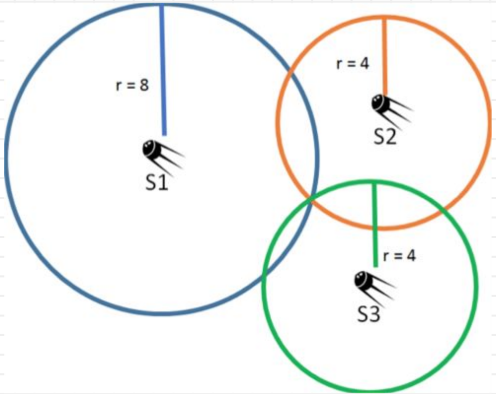
- Satellite:
  - Known, periodic waveforms
  - Know satellite location
- Receiver:
  - Will record the waveform
    - Sum of all shifted beacons
  - Waveform will be shifted from known satellite waveform based on how far it is from satellite (sound takes time to travel)



Let’s go backwards

Assume we know the **distance** between the receiver and every beacon

- Use **lateration** and the satellites’ locations to locate the receiver!
- How many satellites do we need in a 2D world?



How do we get those distances?

Assume we know the **time-delay** (in secs) of every beacon

- Use the **speed that sound travels** through air to get exactly how far our receiver is from every satellite
  - d = v*t

How do we get those time-delays?

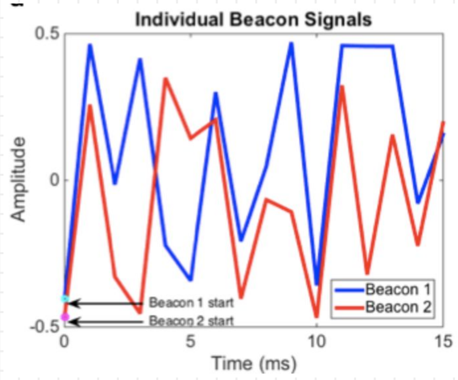
Assume we know how many **samples** it takes for each beacon to arrive at the receiver

- Use the **sampling frequency** of receiver to get the **time-delay**
  - Sampling frequency [samples/sec] - rate at which microphone records samples

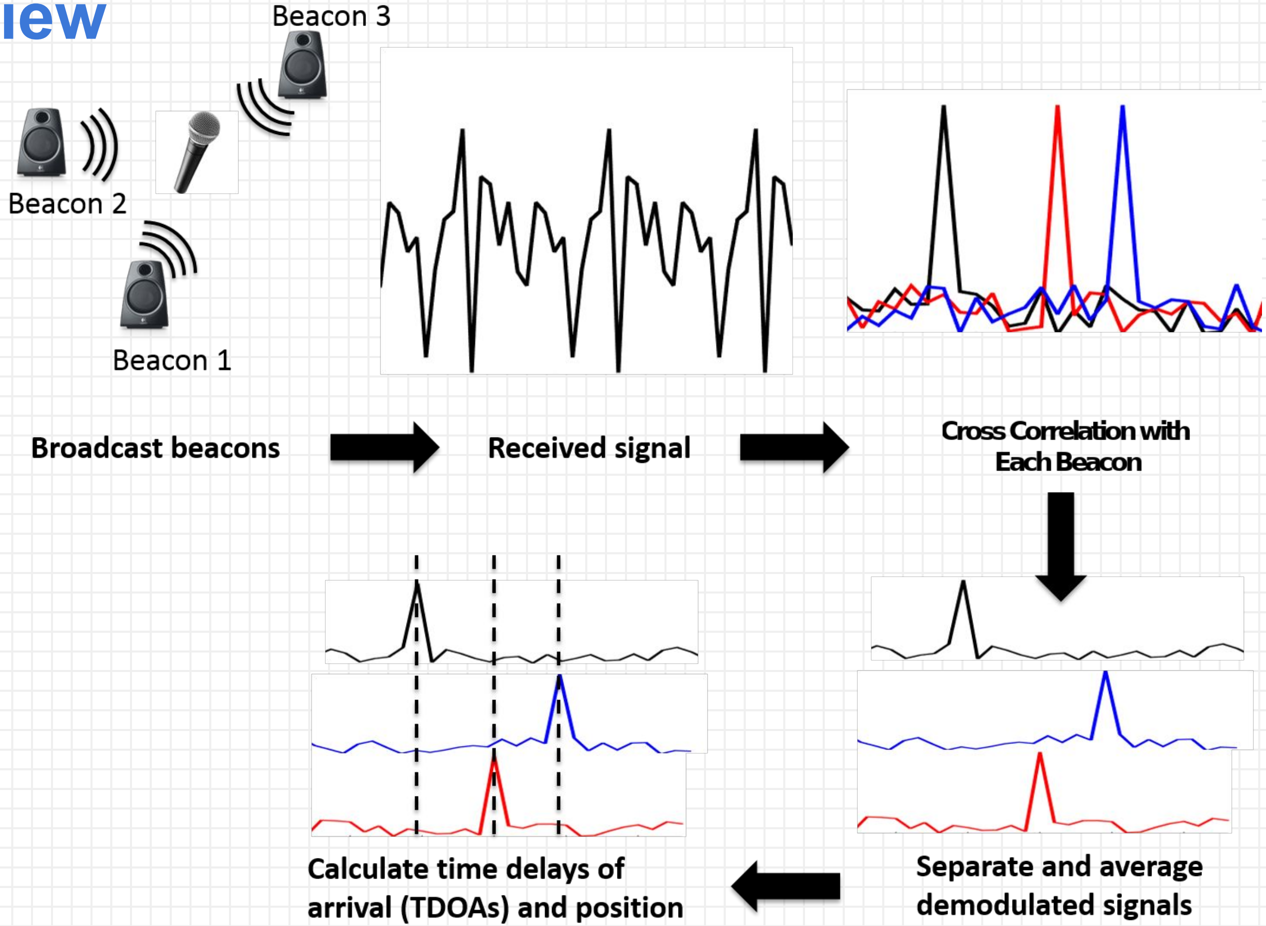
$$\frac{\text{samples}}{f_s} = \frac{\text{samples}}{\frac{samples}{second}} = \text{seconds}$$

How do we get sample delays?

- Receiver's recorded signal is the sum of all the beacon signals
- Need to separate the recorded signal into the individual beacons to see how many samples each is delayed by



Overview


Beacon 3
Beacon 2
Beacon 1
Broadcast beacons
Received signal
Cross Correlation with Each Beacon
Separate and average demodulated signals
Calculate time delays of arrival (TDOAs) and position

Recall: Inner (Dot) product

- A mathematical operation for vectors
- Computes how similar two vectors are

$$
\begin{aligned}
\langle \vec{x}, \vec{y} \rangle \equiv \vec{x} \cdot \vec{y} \equiv \vec{x}^T \vec{y} \\
&= \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \\
&= x_1 y_1 + x_2 y_2 + \cdots + x_n y_n \\
&= \sum_{i=1}^{n} x_i y_i
\end{aligned}
$$

Recall: Inner (Dot) product

$$\langle \vec{x}, \vec{y} \rangle = \|x\| \, \|y\| \cos \theta$$

An alternate form of the dot product

- **Given this expression, with ||x|| = ||y||, when is this expression maximized?**
    - theta = 0
    - vectors point in the <u>**SAME DIRECTION,**</u> so they are the <u>**SAME SIGNAL**</u>

The **bigger** the dot product, the more "**similar**" the two vectors are

Tool: Cross-correlation

$$corr_r(B_A)[k] = \sum_{i=-\infty}^{\infty} r[i] B_A[i-k] \iff$$

In Python:

`cross_correlation(r, B_A)[k]`

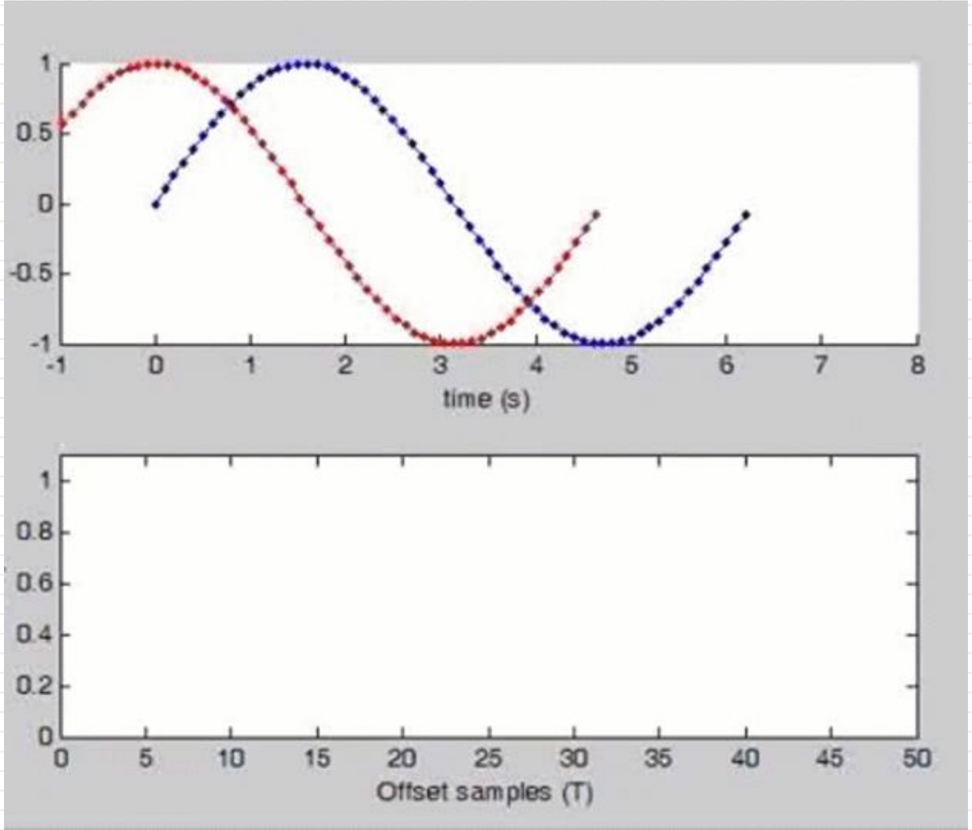
- Mathematical tool for finding similarities between signals
- **Idea**: Computes dot product between r and signal B_A shifted by k samples

- From the previous slide, the peak of the cross-correlation vector tells us which shift amount makes B_A "most similar" to r

Tool: Cross-correlation

- At ~ how many offset samples are the signals most similar?

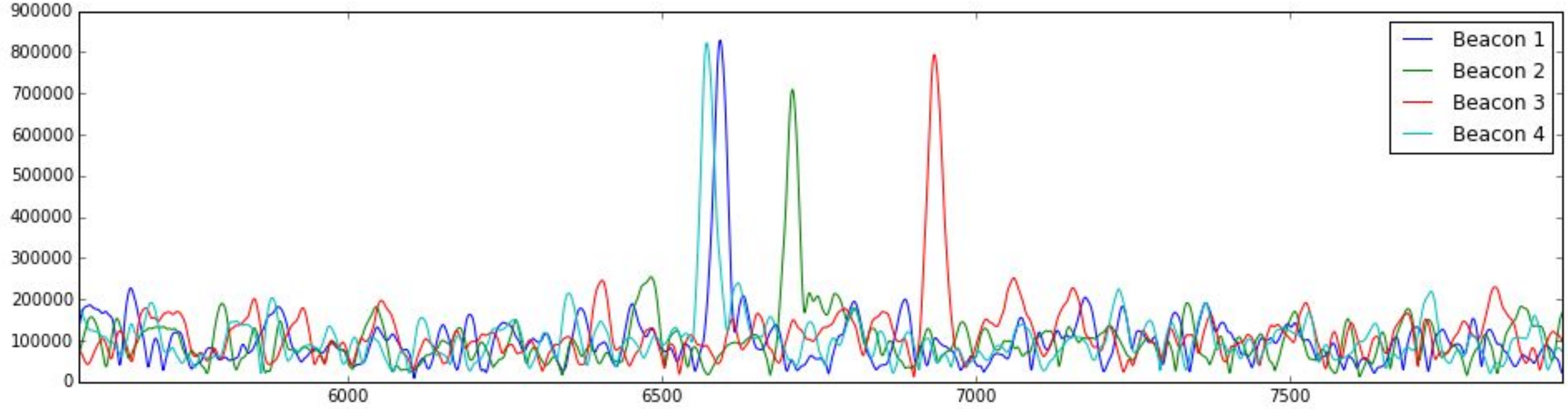
r = blue
B_A = red



Note: zero pad signals to match length

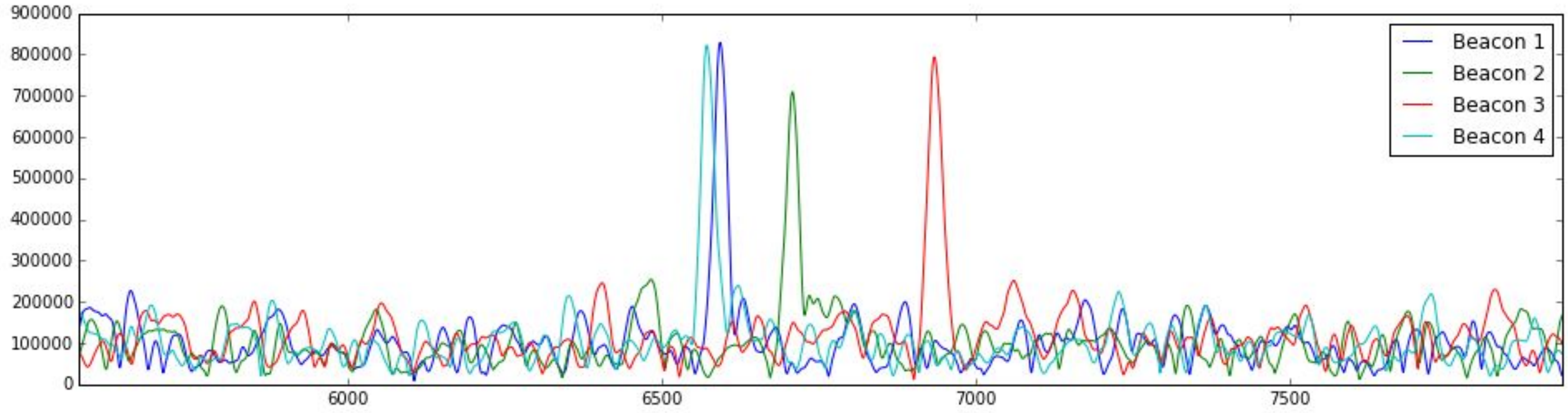
How to use?

- Cross correlating should tell us where each beacon arrived in our recorded signal
- Let's cross-correlate each of the known beacon signals with what we recorded and plot the result



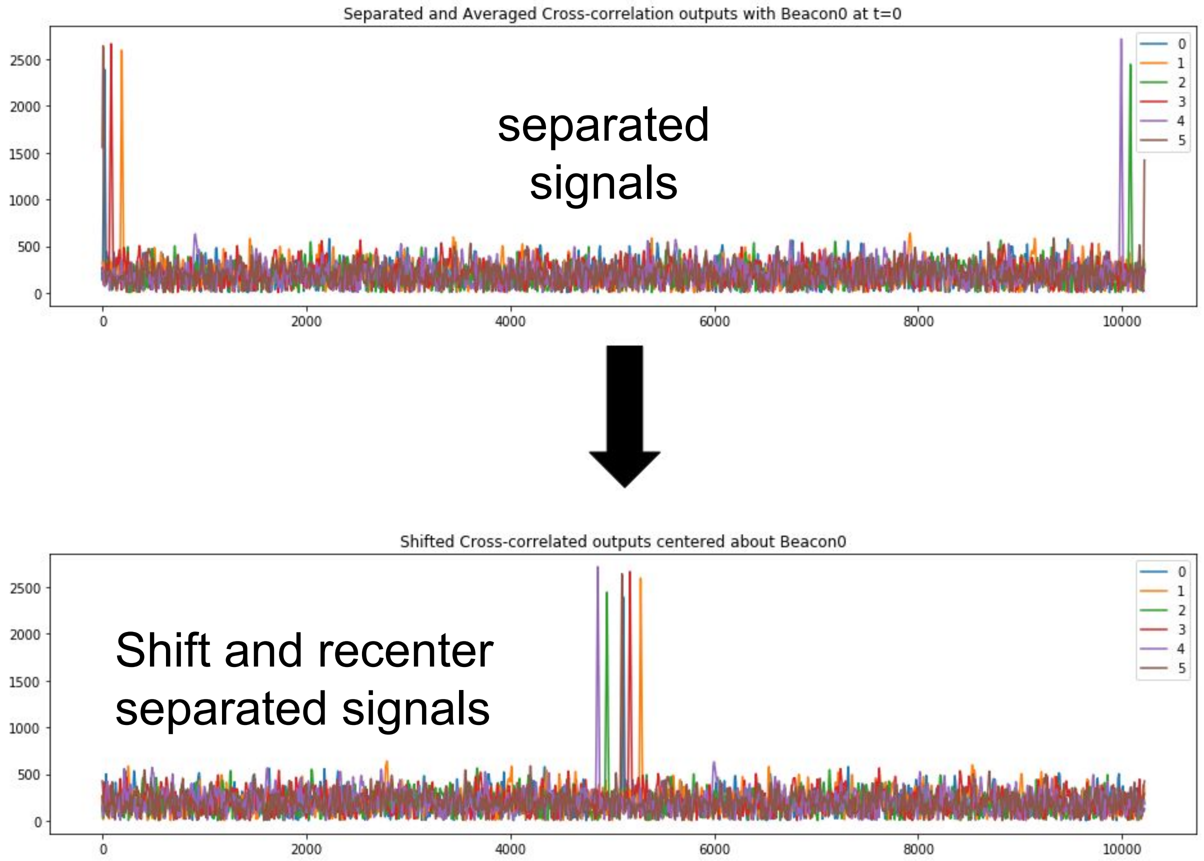
Absolute or relative sample delays?

- We can see peaks where each beacon arrived!
- But notice it only gives us **relative** sample delays
  - Still can’t tell how many absolute samples each beacon is delayed, we don’t know when it was supposed to arrive
- Arbitrarily pick a beacon to be the reference point



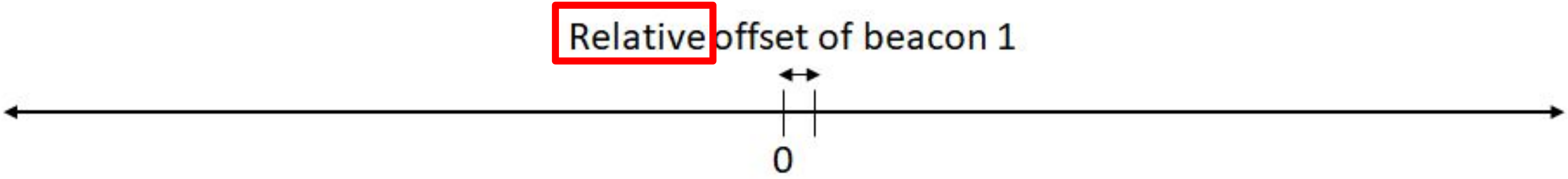
New axis

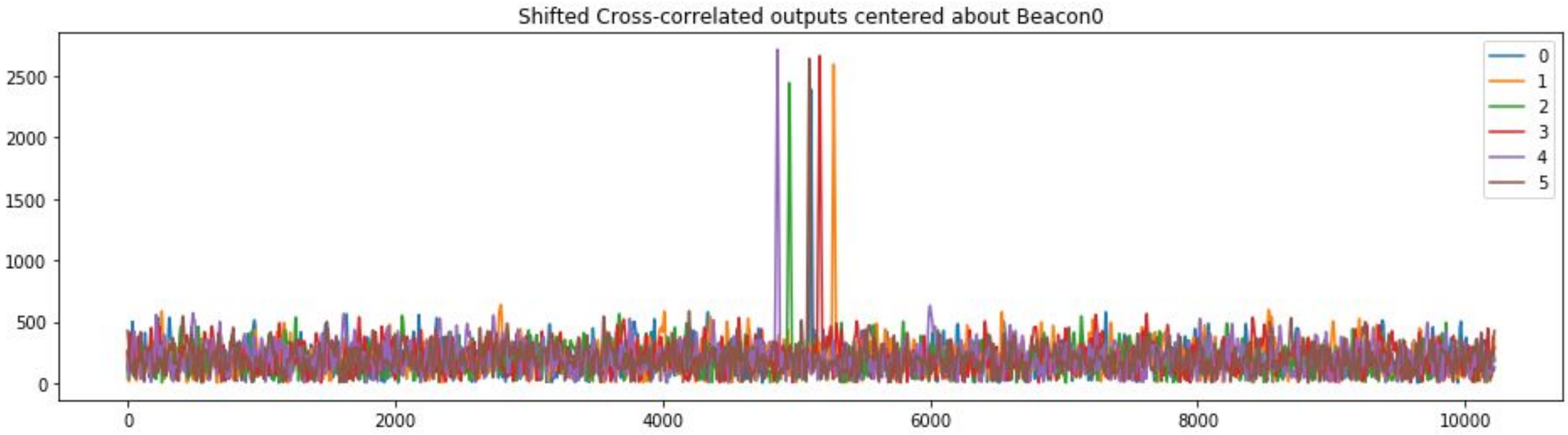
- Let's shift our axis so beacon 0 has a delay of 0
- We could pick any beacon to be the center
  - 0 is arbitrary



New axis

Now beacon 0 is our new “origin” and all computations are relative to the new “0”





Relative Measurements

- Now, we are able compute **relative** sample delays, then **relative** time delays
- How do we get from **relative** time delays to **absolute** distances?
  - With the current set-up, we can’t :(

Additional assumption for APS1

- What if we knew the absolute sample delay of beacon 0?
  - Now we can adjust all our relative measurements to absolute ones!
  - Assume delay_0 is given, then

$$\text{delay}_i = \text{delay}_{i\ \text{relative to } 0} + \text{delay}_0$$

- Then we can use absolute time-delays to get distances then location!

Notes + next lab:

- If we know the absolute sample delay of beacon 0, we can locate the receiver
  - Note that this the same as telling you exactly how far the receiver is from satellite 0
- This week, this value will be given to you
- Find out how to get around this assumption in APS 2!
- Don't forget to turn in PDF and complete checkoff quiz on Gradescope