Evaluating Methods for Radio Frequency Based Positioning of Ultimate Frisbee Players

Bem Abebayehu, Cullen Baker, Marshall Johnson,
Sam Diana, Selma Vangstein, Will Shrestha

Introduction

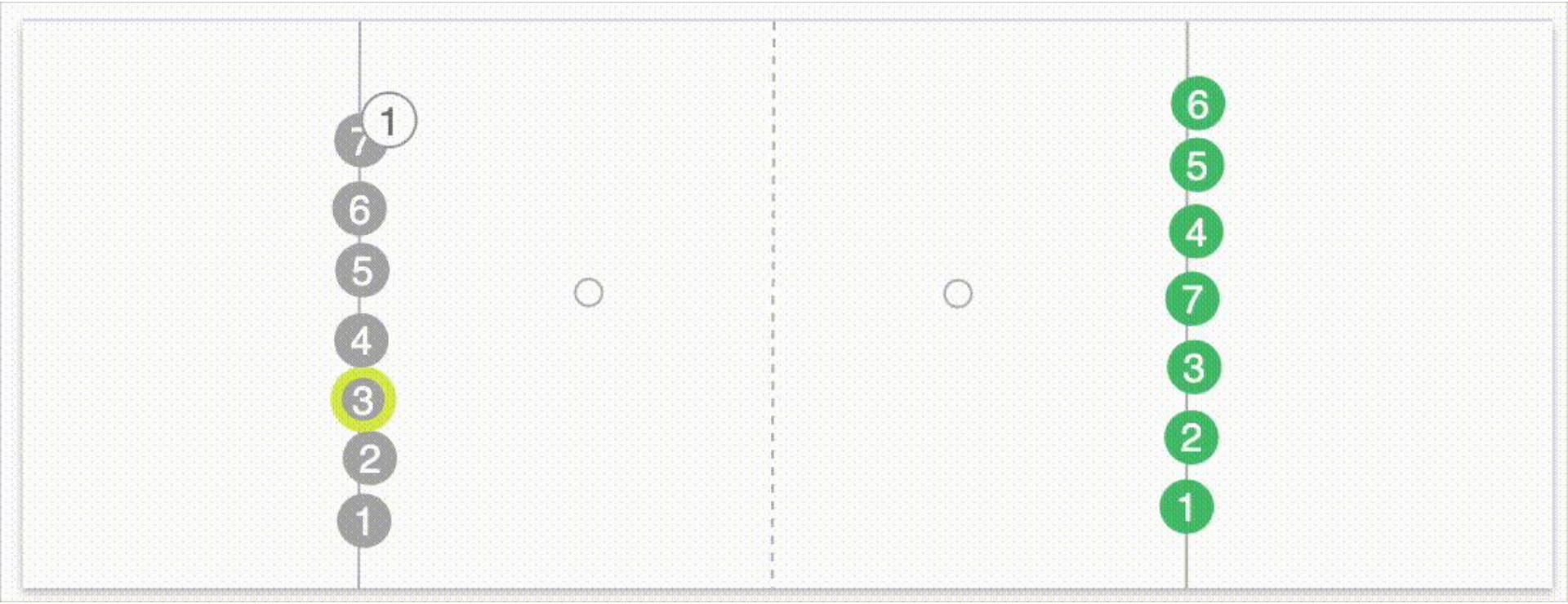
- Accurate player tracking is essential
- Less funding and attention than mainstream sports
- Advanced position-tracking technologies are inaccessible

Introduction

Current Workaround Methods

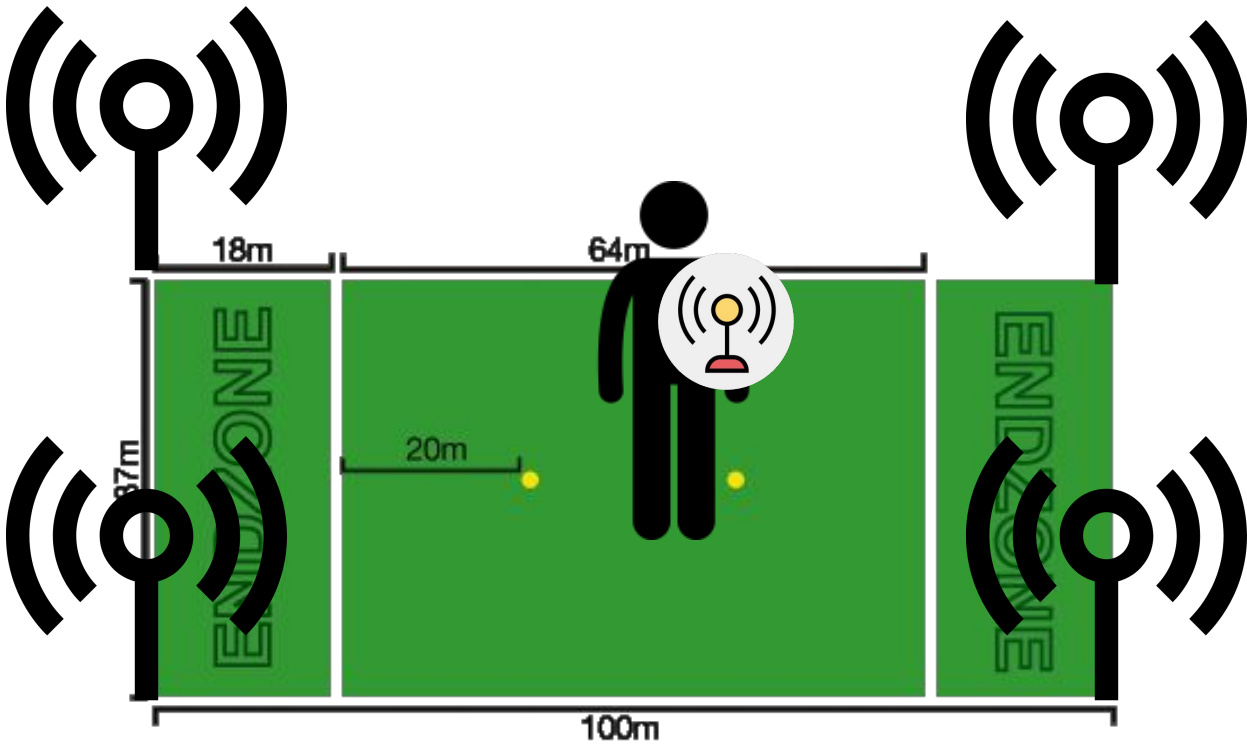
- Drone
  - Expensive, low battery
- Video Camera
  - Field of view issues
- GPS Apps
  - Inaccurate

End Goal



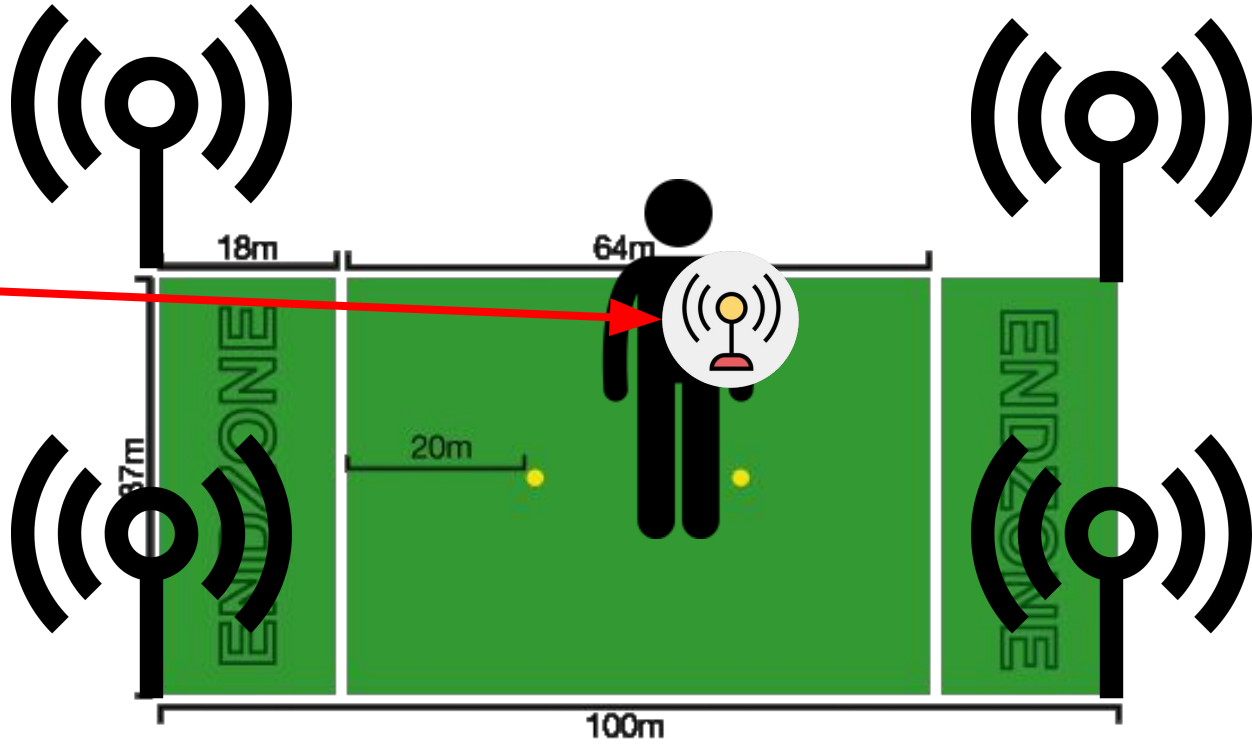
Implementation

- **Beacons** at known points
- **Wearable** knows distances to each beacon
- Use geometry to calculate **position**



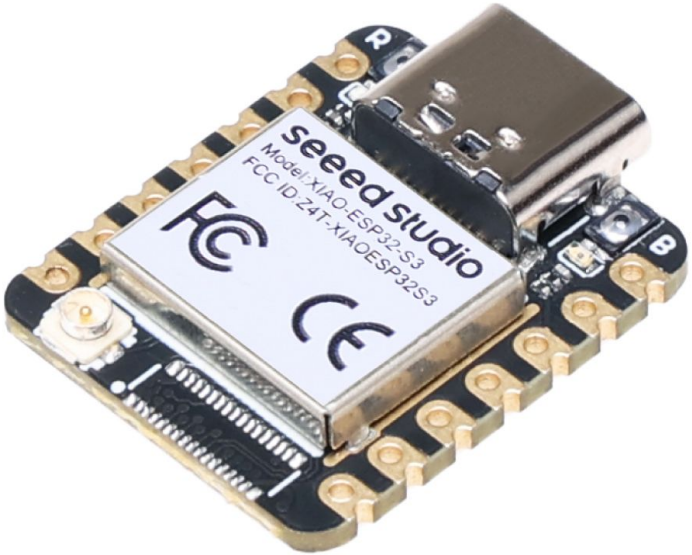
Implementation

- But what *are* these devices?



They’re microcontrollers!

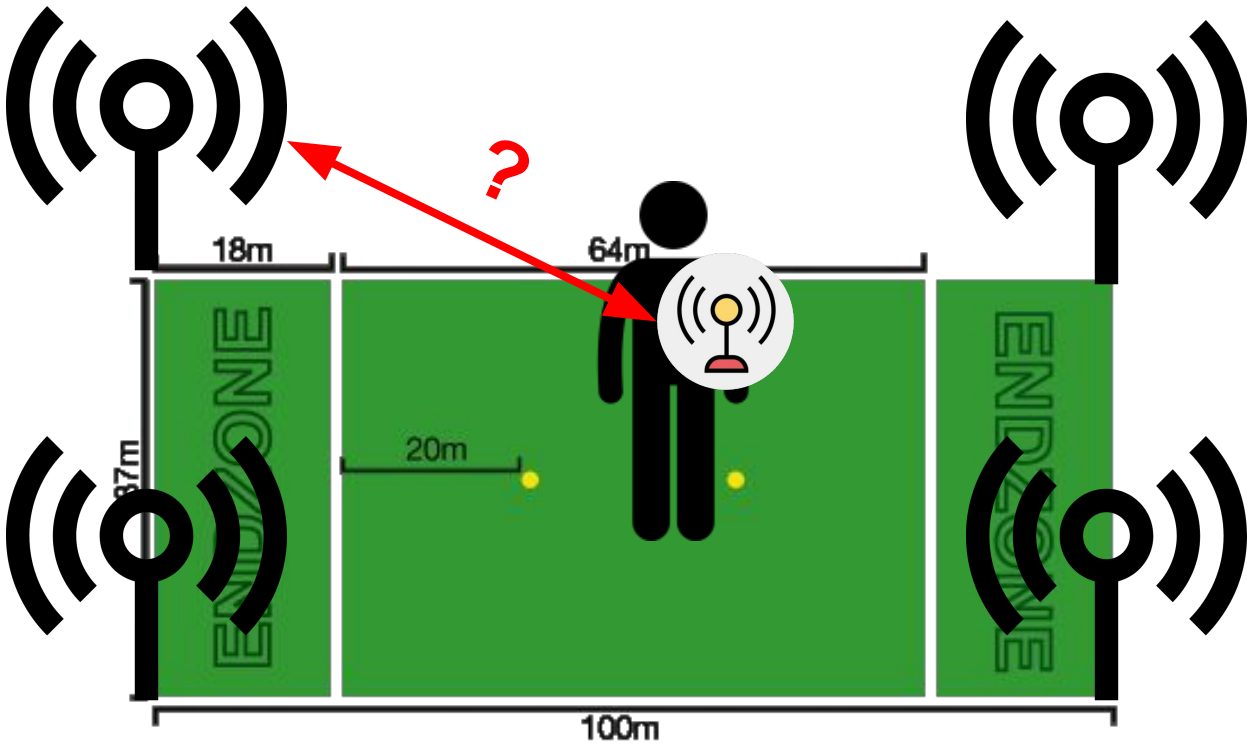
- Small, single board computers
  - Using ESP32s
  - WiFi and Bluetooth capable devices
  - Easy to add sensors



A xiao esp32s3 module - this was our starting point

Getting Distances

- Need to find **distance** between **wearable** and each **beacon**



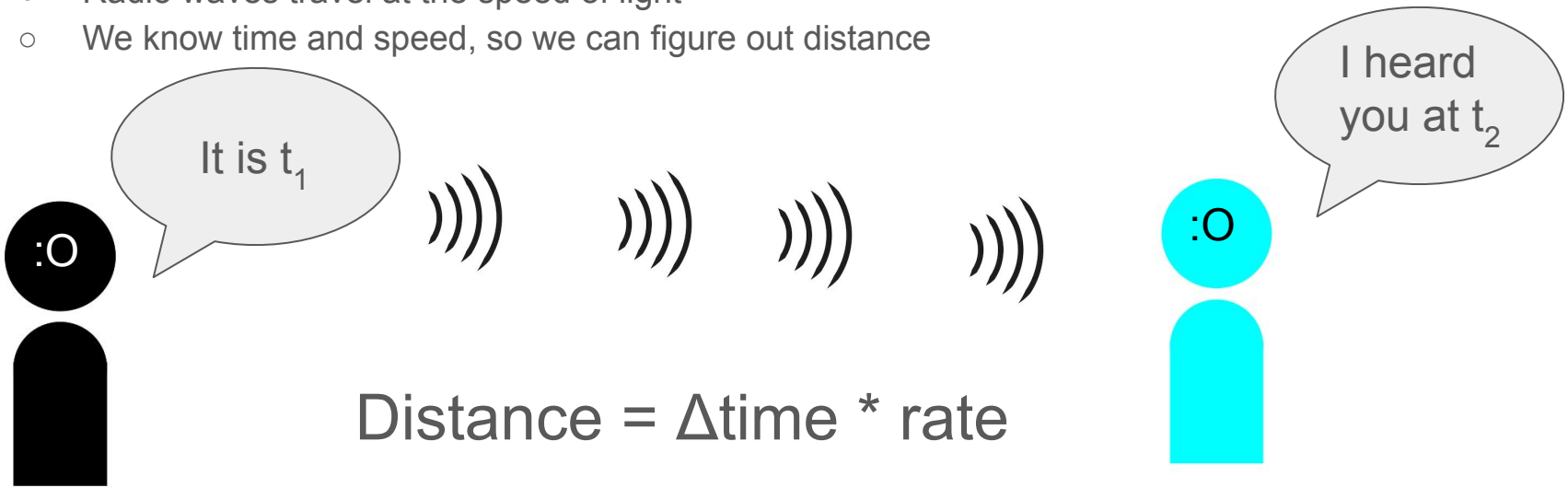
RSSI

- **R**eceived **S**ignal **S**trength **I**ndicator
- Used for several different radio based technologies
  - Used w/ Bluetooth and WiFi
- Implementation varies by manufacturer

RSSI	Signal Strength
> -70 dBm	Excellent
-70 dBm to -85 dBm	Good
-86 dBm to -100 dBm	Fair
< -100 dBm	Poor
-110 dBm	No signal

Time of Flight approach

- High level: record how long it takes for a message to be sent back and forth
  - Radio waves travel at the speed of light
  - We know time and speed, so we can figure out distance

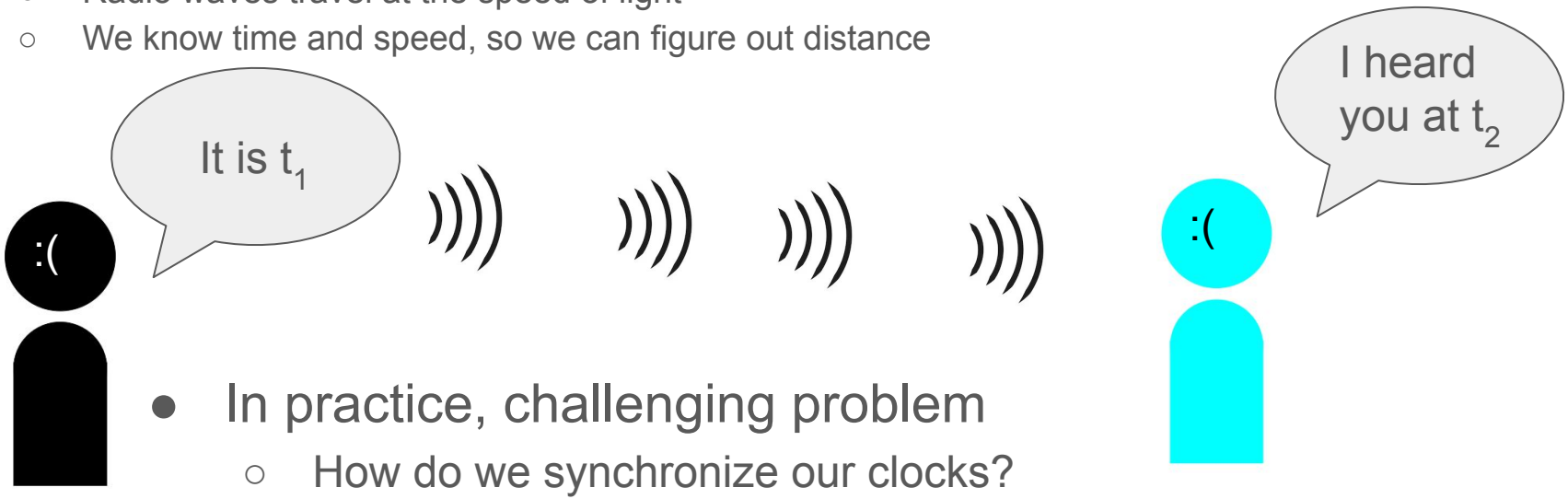


$$\text{Distance} = \Delta\text{time} * \text{rate}$$

$$d = (t_2 - t_1) * c$$

Time of Flight approach

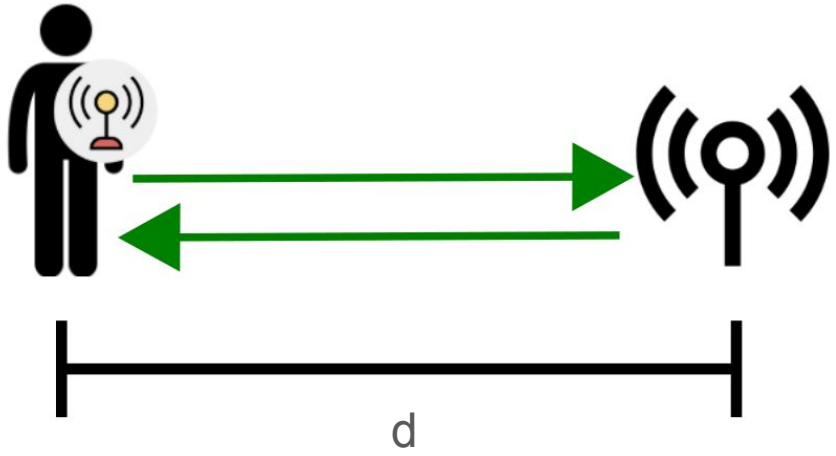
- High level: record how long it takes for a message to be sent back and forth
  - Radio waves travel at the speed of light
  - We know time and speed, so we can figure out distance



- In practice, challenging problem
  - How do we synchronize our clocks?
    - Light is fast - time difference is tiny!

Two way ranging

- High level: send messages **back and forth**
  - We're interested in finding **R**ound **T**rip **T**ime
  - Keep track of when you send a message
  - Record relative time differences

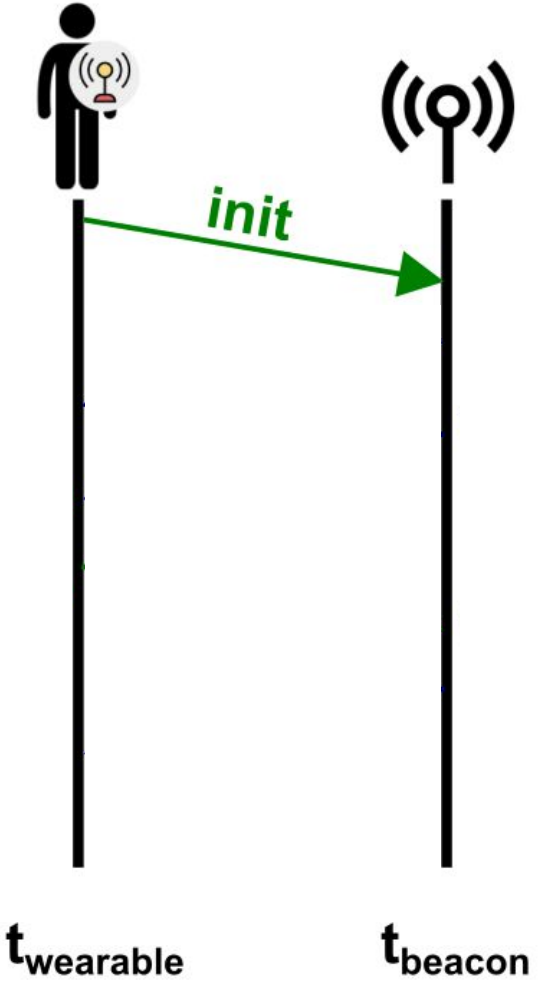


$$d = (t_{rtt} * c) / 2$$

- IEEE 802.11mc for wifi
  - Dubbed wifi FTM
- IEEE 802.15.4-2011 for UWB
  - Dubbed UWB RTLS

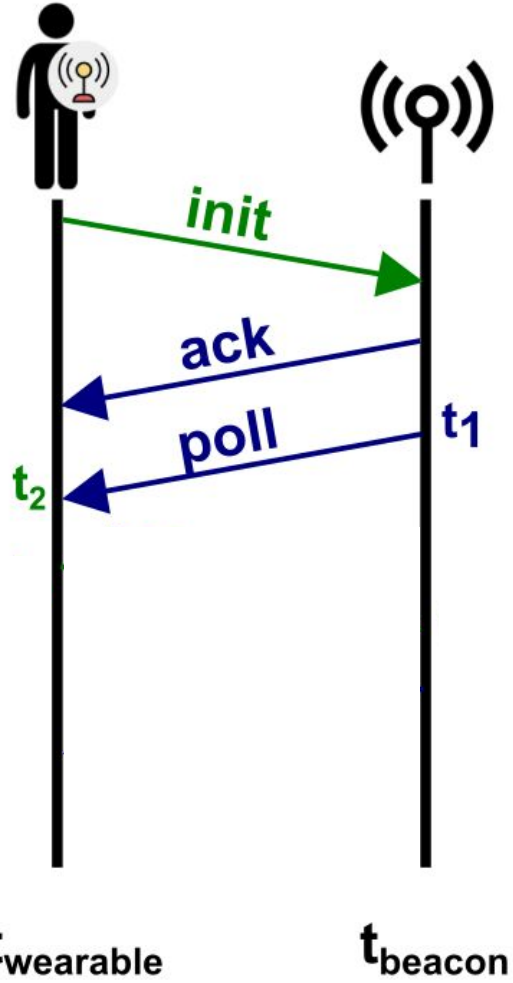
Two way ranging

- **Init**: Initialize ranging
  - Both parties get ready to do a ranging exchange



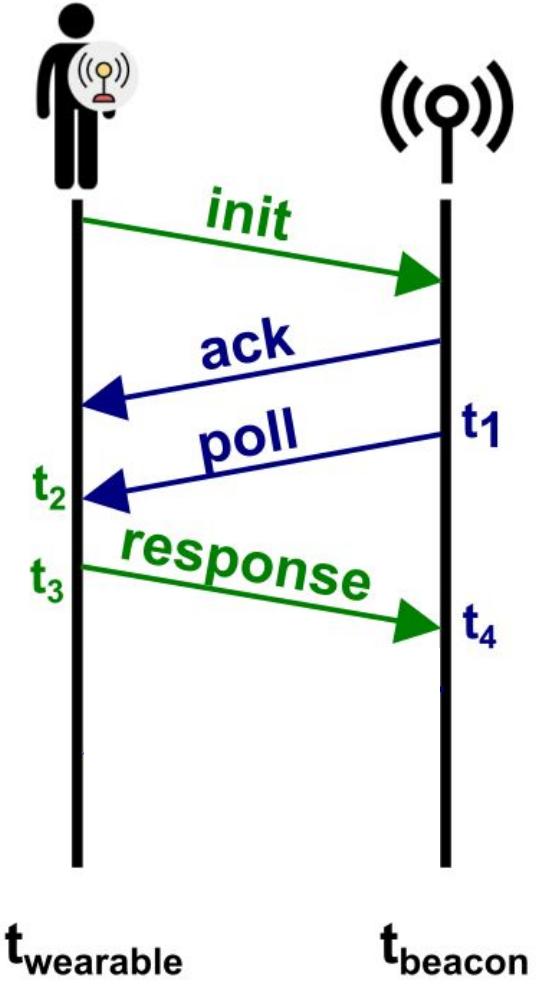
Two way ranging

- **Ack**: Acknowledged ranging request
  - Heard your message, let's do this
- **Poll**: Polling message
  - First message of round trip



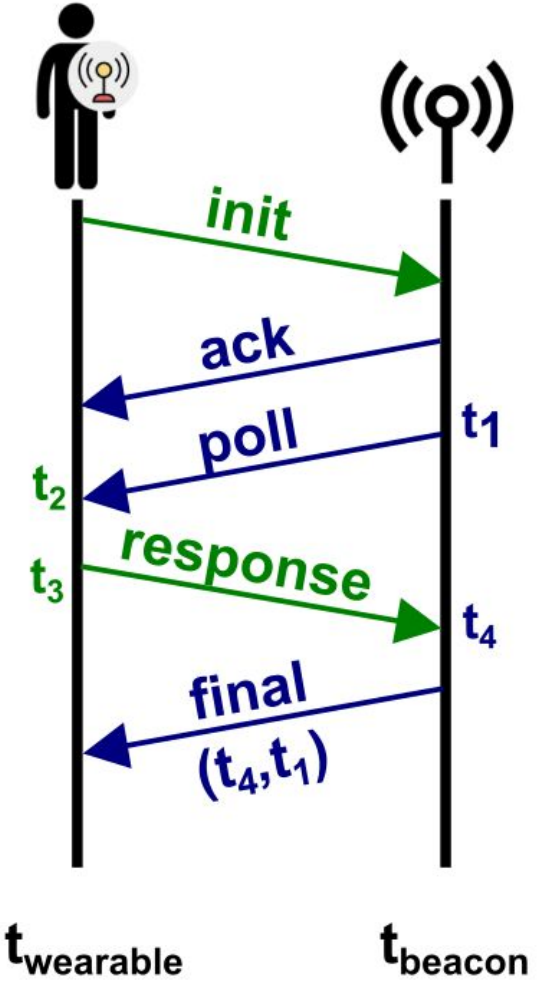
Two way ranging

- **Response**: follow up to poll
  - Other half of round trip
  - Again, timing is recorded



Two way ranging

- **Final**: last message
  - Gives the wearable all the data it needs
  - Timing not critical, so long as it arrives

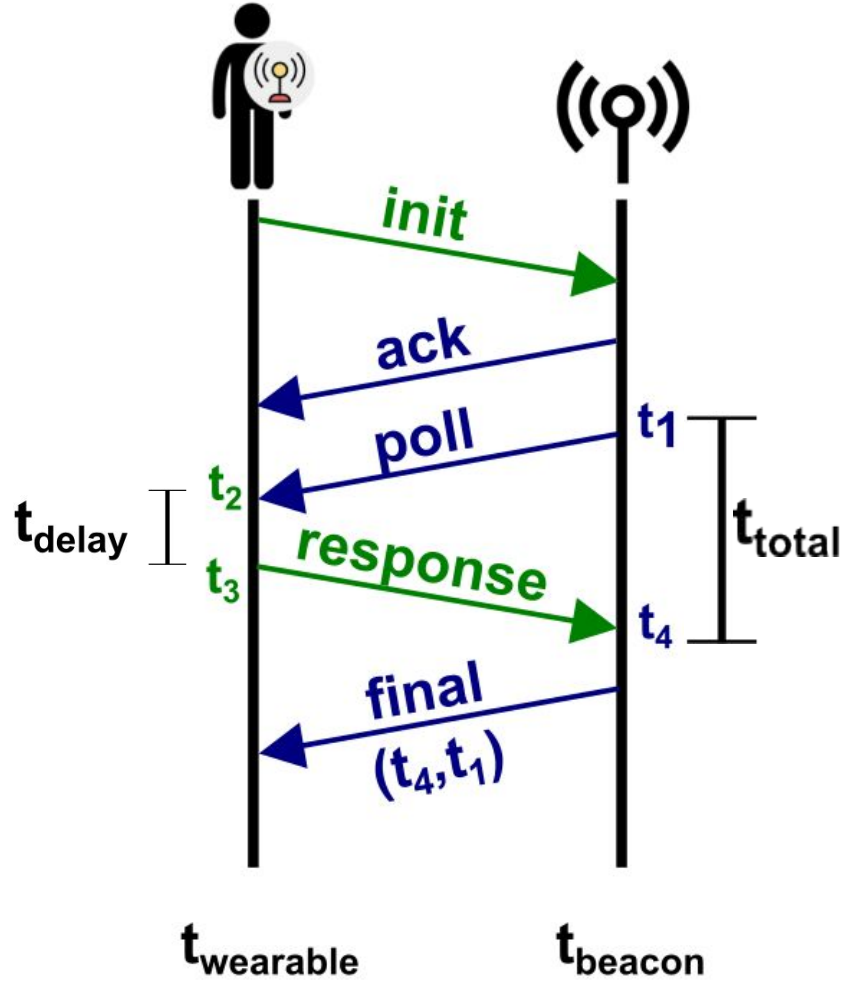


Two way ranging

$$t_{total} = t_4 - t_1$$

However, this includes delay between t_2 and t_3

So let's just subtract it!



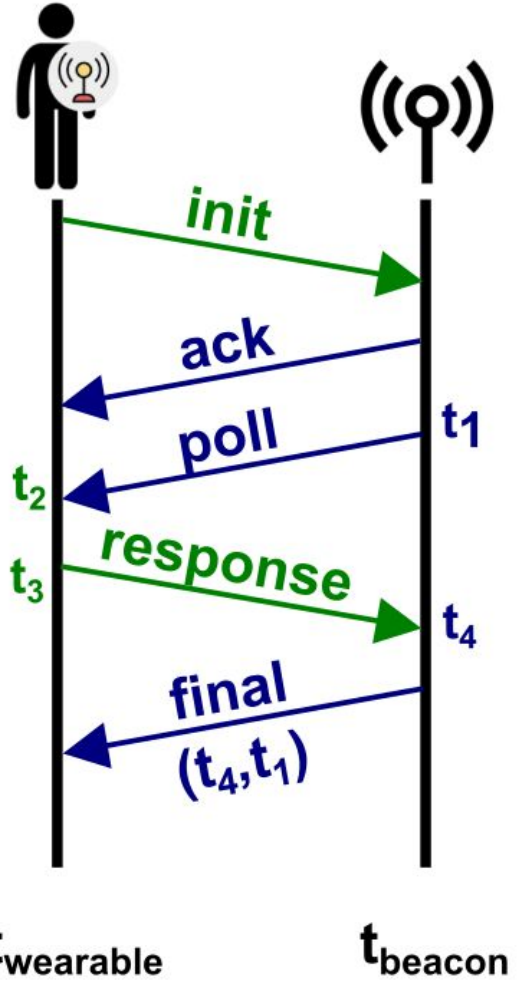
Two way ranging

$$t_{total} = t_4 - t_1$$

$$t_{rtt} = t_4 - t_1 - (t_3 - t_2)$$

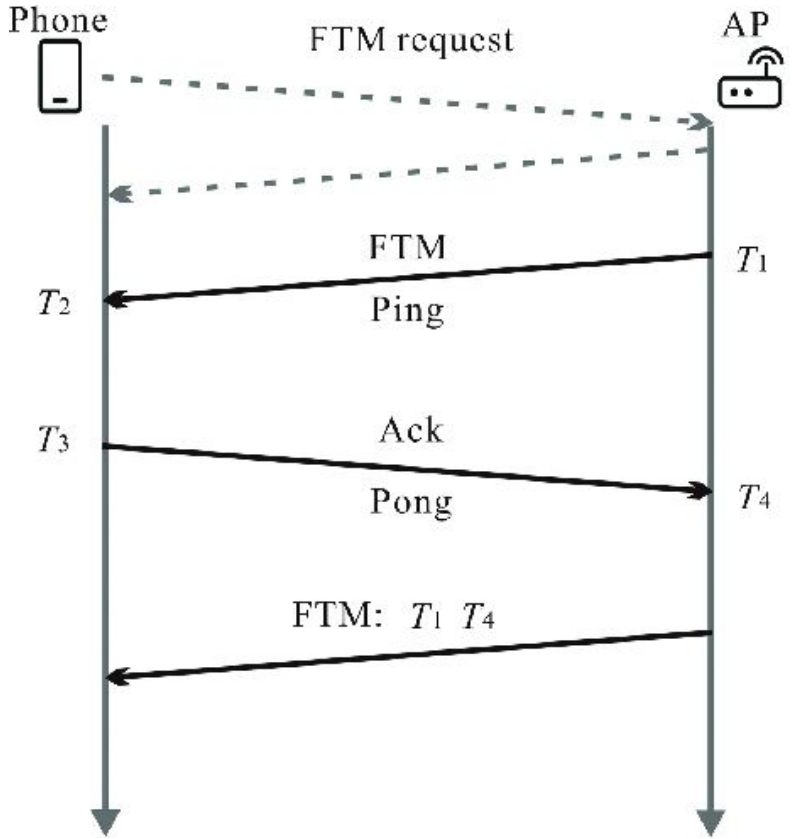
Left with **R**ound **T**rip **T**ime

- Aka: time for message to go back and forth (2*d)



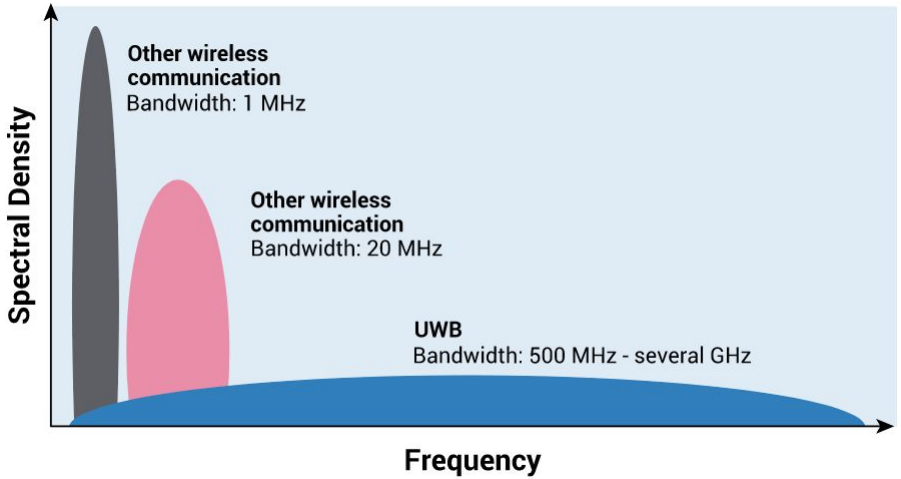
FTM

- **F**ine **T**iming **M**easurement
- **R**ound **T**rip **T**iming (RTT) using WiFi
- IEEE 802.11mc standard



UWB

- Much wider frequency range
  - Requires different hardware, can't use built-in wifi hardware like FTM
- Wider waves, less vulnerable to obstructions
- Rapid bursts allow for higher accuracy
- Lower power



Implementation: what we tried

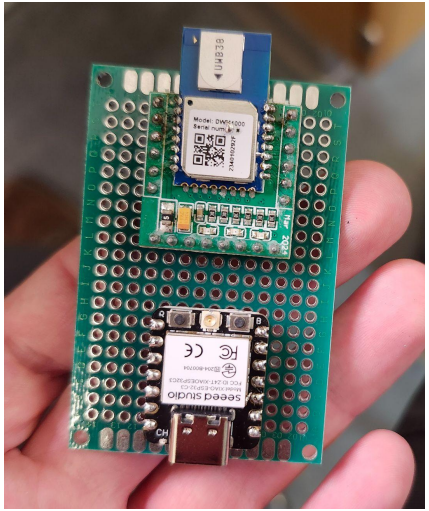
	Bluetooth	**Wi-Fi**	**Ultrawideband**
Cost:	🤑 ($7-$12)	🤑($7-$12)	🤑🤑🤑($40-50 prebuilt)
Range:	📏	📏📏	📏📏📏
Accuracy:	🎯	🎯🎯	🎯🎯🎯

Board Prototyping

Two main tasks:

- Make FTM better
- Make UWB cheaper

We tried a lot

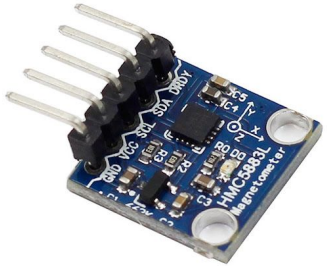


For FTM:

- Compass modules
- Accelerometers

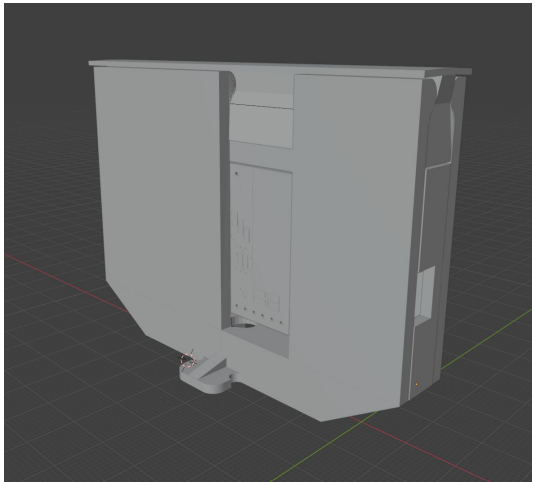
For UWB:

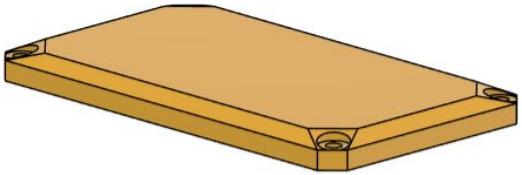
- Custom, cheaper UWB modules
- Pre Built prototyping boards

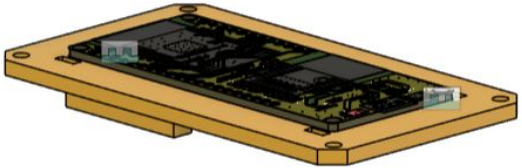


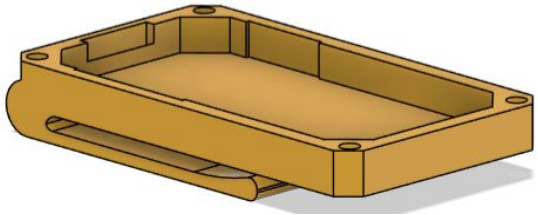
Wearable and Mount Prototyping

We also tried different housing methods, trying to keep the signal unobstructed, the modules protected, and the players comfortable.



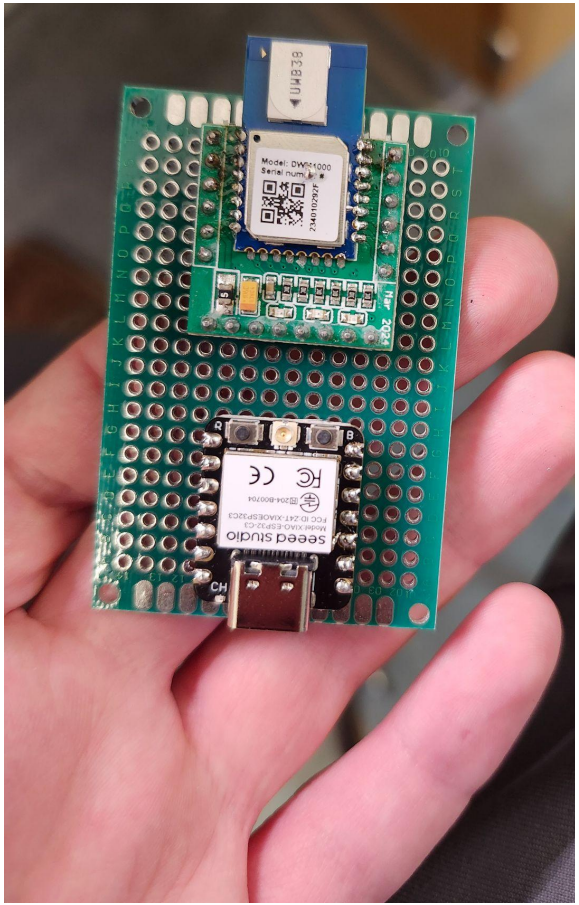






Custom UWB modules

- Up until 3rd week this term, UWB was a deprioritized side project
- Chose dwm1000 uwb modules and Xiao esp32 c3 arduinos
- Had numerous issues with these two modules together
- If I could go back in time, arduino nano or s3. We weren't able to figure out why they didn't seem to work with the c3, but there is existing documentation for these modules

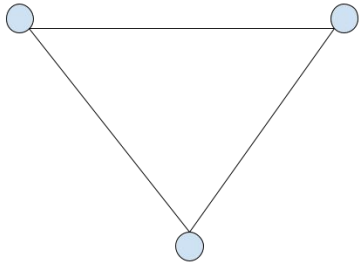


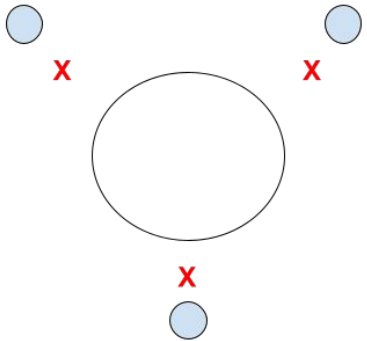
ESP32 UWB Prototyping Boards

- Due to the difficulties with custom UWB modules and minimal time left, we decided to order ESP 32 UWB modules
- They worked immediately out of the box
- Software issues remain
- Price is too high

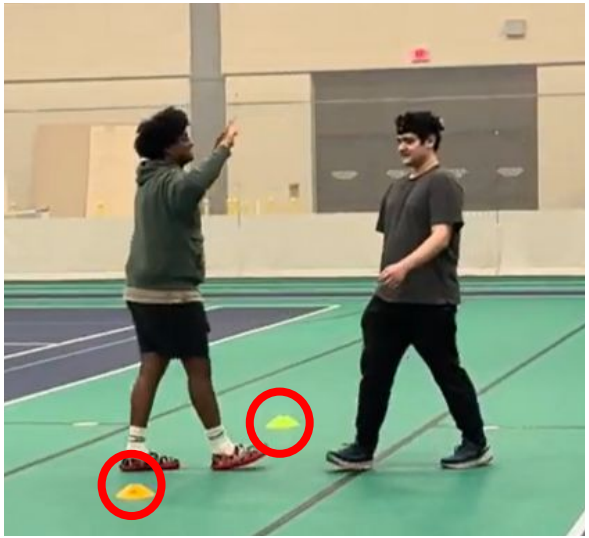
Wearables and Mounts

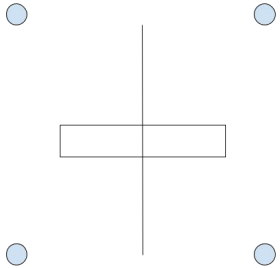
Fall Term Testing

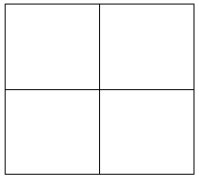




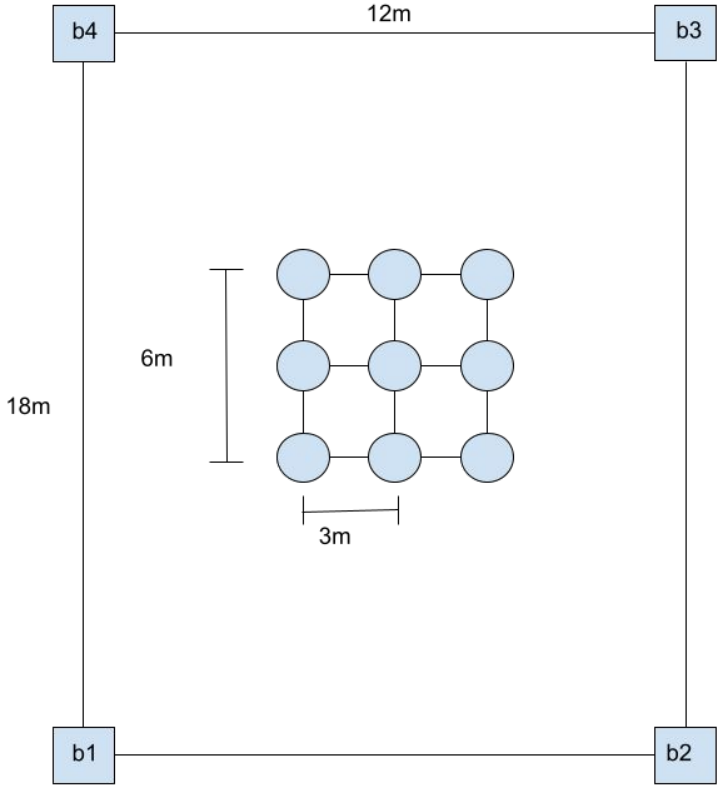
Winter Term Testing







Ground Truth



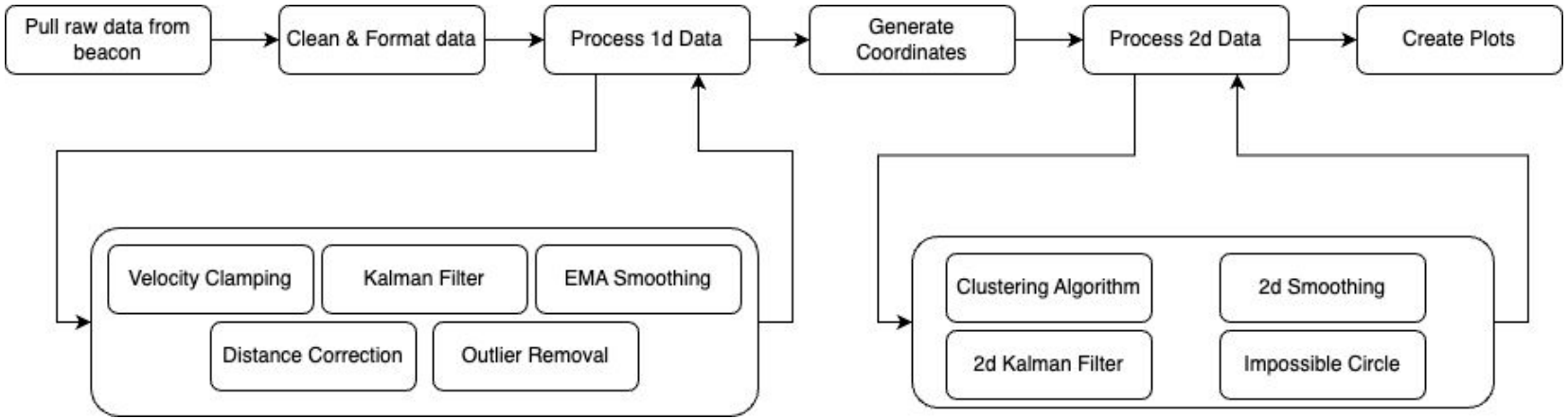
Rec Testing Layout

Data Processing

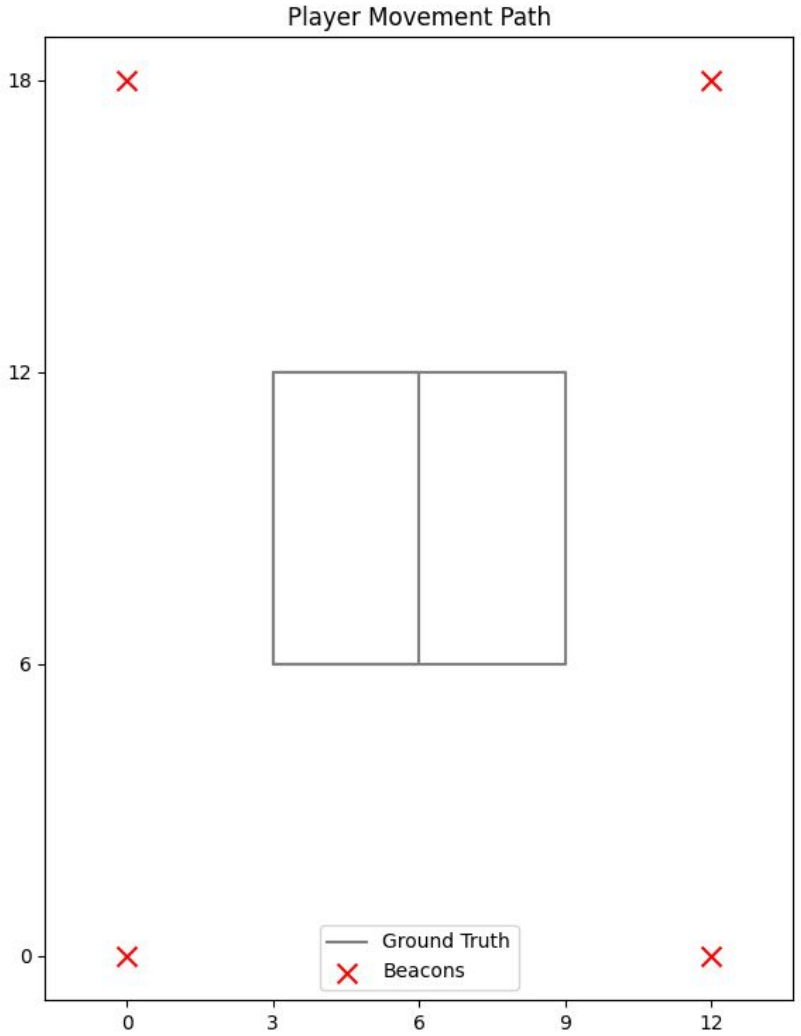
Problem: no way for every group member to quickly analyze data from a test

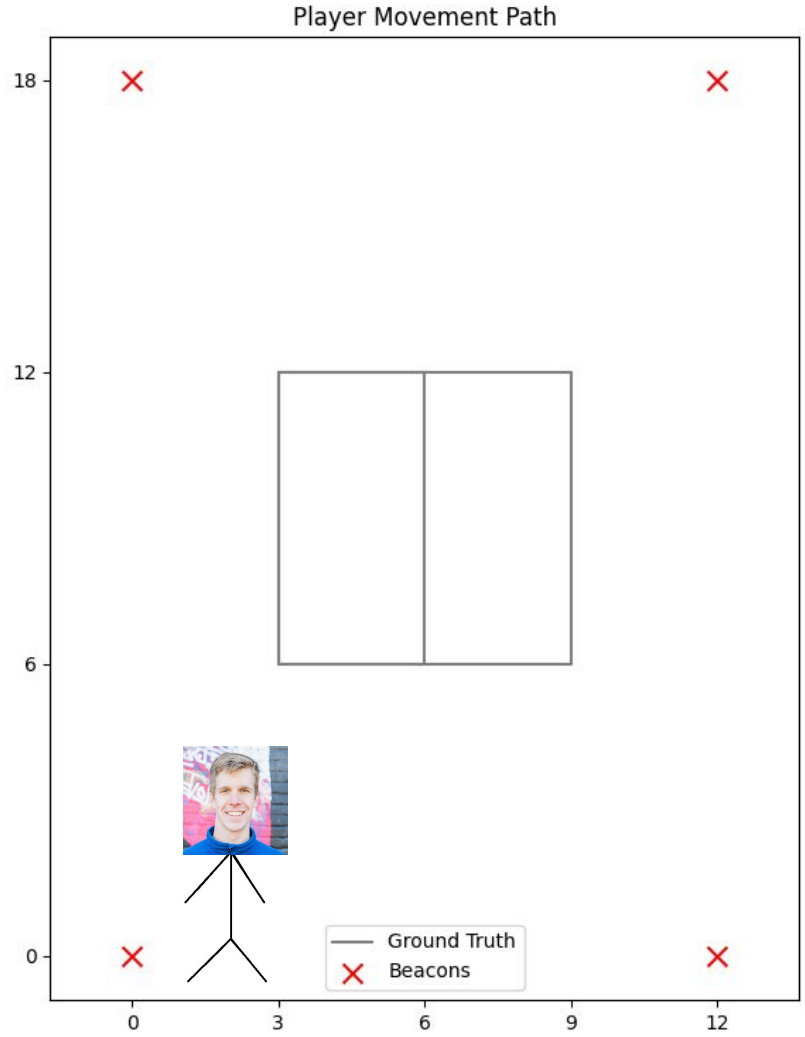
Why?: Data is noisy and we needed a way to clean it

Solution: “Post Processing Pipeline”

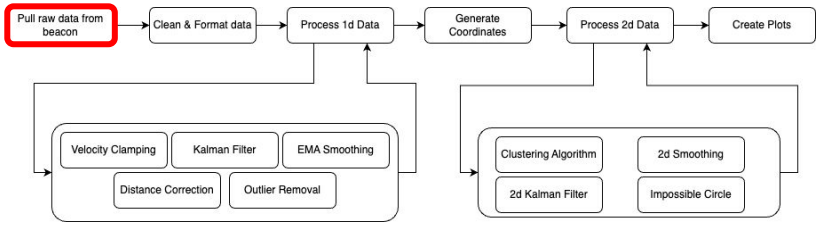


Processing Walkthrough




Player Movement Path
18
12
6
0
0
3
6
9
12
Ground Truth
Beacons

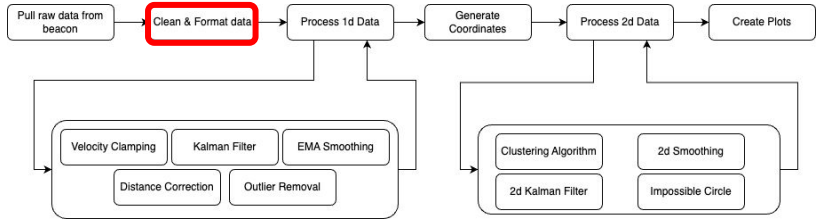
Beacon Data



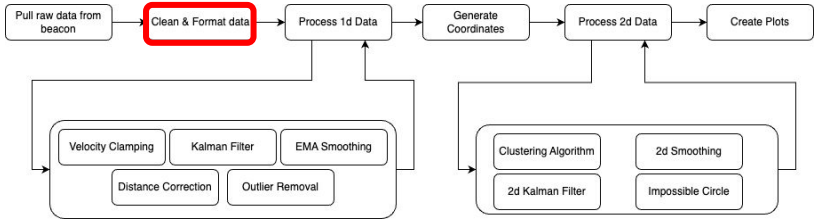

```
4
5   [21:48:26.691] a,134261,15.690000,20.800000,23.560000,20.370000,-0.620000,-4.430000,10.390000
6   [21:48:26.741] a,134261,15.690000,20.800000,23.560000,20.370000,-0.620000,-4.430000,10.390000
7   [21:48:26.791] a,134363,15.580000,20.830000,23.560000,20.400000,-0.540000,-4.430000,10.900000
8   [21:48:26.842] a,134363,15.580000,20.830000,23.560000,20.400000,-0.540000,-4.430000,10.900000
9   [21:48:26.892] a,134465,15.650000,20.860000,23.560000,20.360000,-0.940000,-3.880000,10.980000
10  [21:48:26.942] a,134465,15.650000,20.860000,23.560000,20.360000,-0.940000,-3.880000,10.980000
11  [21:48:26.992] a,134567,15.650000,20.860000,23.560000,20.360000,-0.860000,-3.800000,11.250000
12  [21:48:27.043] a,134567,15.650000,20.860000,23.560000,20.360000,-0.860000,-3.800000,11.250000
13  [21:48:27.093] a,134669,15.650000,20.860000,23.560000,20.360000,-1.720000,-4.270000,11.370000
14  [21:48:27.144] a,134669,15.650000,20.860000,23.560000,20.360000,-1.720000,-4.270000,11.370000
15  [21:48:27.194] a,134771,15.650000,20.840000,23.560000,20.360000,-0.980000,-4.110000,10.860000
16  [21:48:27.245] a,134771,15.650000,20.840000,23.560000,20.360000,-0.980000,-4.110000,10.860000
17  [21:48:27.295] a,134874,15.610000,20.840000,23.560000,20.320000,-1.920000,-3.880000,11.060000
```

```
298
299   [10:14:40.937] NO_RESPONSE
300
301   [10:14:42.437] NO_RESPONSE
```

Cleaned Data



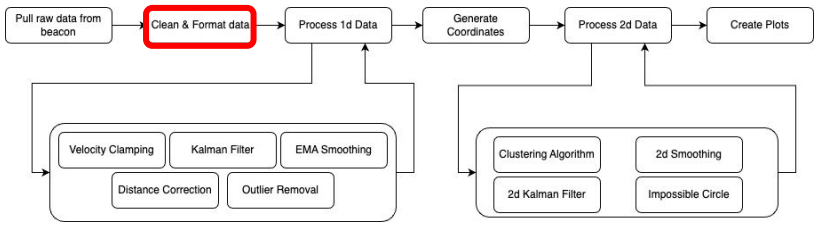

```
timestamp,playerid,wearabletimestamp,b1d,b2d,b3d,b4d,xa,ya,za
21:48:26.677,a,134159,15.690000,20.800000,23.560000,20.370000,-1.010000,-4.470000,10.940000
21:48:26.677,a,134159,15.690000,20.800000,23.560000,20.370000,-1.010000,-4.470000,10.940000
21:48:26.691,a,134261,15.690000,20.800000,23.560000,20.370000,-0.620000,-4.430000,10.390000
21:48:26.741,a,134261,15.690000,20.800000,23.560000,20.370000,-0.620000,-4.430000,10.390000
21:48:26.791,a,134363,15.580000,20.830000,23.560000,20.400000,-0.540000,-4.430000,10.900000
21:48:26.842,a,134363,15.580000,20.830000,23.560000,20.400000,-0.540000,-4.430000,10.900000
21:48:26.892,a,134465,15.650000,20.860000,23.560000,20.360000,-0.940000,-3.880000,10.980000
21:48:26.942,a,134465,15.650000,20.860000,23.560000,20.360000,-0.940000,-3.880000,10.980000
21:48:26.992,a,134567,15.650000,20.860000,23.560000,20.360000,-0.860000,-3.800000,11.250000
21:48:27.043,a,134567,15.650000,20.860000,23.560000,20.360000,-0.860000,-3.800000,11.250000
21:48:27.093,a,134669,15.650000,20.860000,23.560000,20.360000,-1.720000,-4.270000,11.370000
21:48:27.144,a,134669,15.650000,20.860000,23.560000,20.360000,-1.720000,-4.270000,11.370000
21:48:27.194,a,134771,15.650000,20.840000,23.560000,20.360000,-0.980000,-4.110000,10.860000
21:48:27.245,a,134771,15.650000,20.840000,23.560000,20.360000,-0.980000,-4.110000,10.860000
21:48:27.295,a,134874,15.610000,20.840000,23.560000,20.320000,-1.920000,-3.880000,11.060000
21:48:27.346,a,134874,15.610000,20.840000,23.560000,20.320000,-1.920000,-3.880000,11.060000
```



Cleaned Data

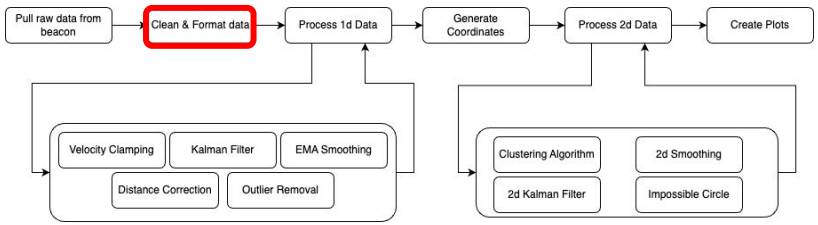
```
timestamp,playerid,wearabletimestamp,b1d,b2d,b3d,b4d,xa,ya,za
21:48:26.677,a,134159,15.690000,20.800000,23.560000,20.370000,-1.010000,-4.470000,10.940000
21:48:26.677,a,134159,15.690000,20.800000,23.560000,20.370000,-1.010000,-4.470000,10.940000
21:48:26.691,a,134261,15.690000,20.800000,23.560000,20.370000,-0.620000,-4.430000,10.390000
21:48:26.741,a,134261,15.690000,20.800000,23.560000,20.370000,-0.620000,-4.430000,10.390000
21:48:26.791,a,134363,15.580000,20.830000,23.560000,20.400000,-0.540000,-4.430000,10.900000
21:48:26.842,a,134363,15.580000,20.830000,23.560000,20.400000,-0.540000,-4.430000,10.900000
21:48:26.892,a,134465,15.650000,20.860000,23.560000,20.360000,-0.940000,-3.880000,10.980000
21:48:26.942,a,134465,15.650000,20.860000,23.560000,20.360000,-0.940000,-3.880000,10.980000
21:48:26.992,a,134567,15.650000,20.860000,23.560000,20.360000,-0.860000,-3.800000,11.250000
21:48:27.043,a,134567,15.650000,20.860000,23.560000,20.360000,-0.860000,-3.800000,11.250000
21:48:27.093,a,134669,15.650000,20.860000,23.560000,20.360000,-1.720000,-4.270000,11.370000
21:48:27.144,a,134669,15.650000,20.860000,23.560000,20.360000,-1.720000,-4.270000,11.370000
21:48:27.194,a,134771,15.650000,20.840000,23.560000,20.360000,-0.980000,-4.110000,10.860000
21:48:27.245,a,134771,15.650000,20.840000,23.560000,20.360000,-0.980000,-4.110000,10.860000
21:48:27.295,a,134874,15.610000,20.840000,23.560000,20.320000,-1.920000,-3.880000,11.060000
21:48:27.346,a,134874,15.610000,20.840000,23.560000,20.320000,-1.920000,-3.880000,11.060000
```

Cleaned Data



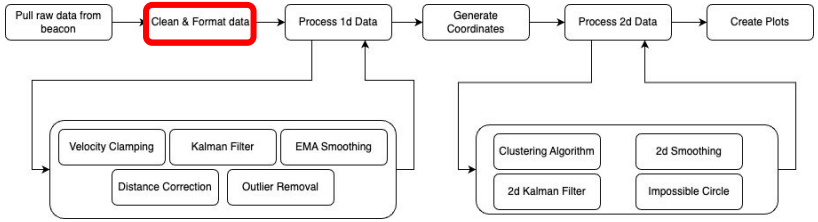
	timestamp	playerid	wearabletimestamp	b1d	b2d	b3d	b4d	xa	ya	za
2	21:48:26.677	a	134159	15.690000	20.800000	23.560000	20.370000	-1.010000	-4.470000	10.940000
3	21:48:26.677	a	134159	15.690000	20.800000	23.560000	20.370000	-1.010000	-4.470000	10.940000
4	21:48:26.691	a	134261	15.690000	20.800000	23.560000	20.370000	-0.620000	-4.430000	10.390000
5	21:48:26.741	a	134261	15.690000	20.800000	23.560000	20.370000	-0.620000	-4.430000	10.390000
6	21:48:26.791	a	134363	15.580000	20.830000	23.560000	20.400000	-0.540000	-4.430000	10.900000
7	21:48:26.842	a	134363	15.580000	20.830000	23.560000	20.400000	-0.540000	-4.430000	10.900000
8	21:48:26.892	a	134465	15.650000	20.860000	23.560000	20.360000	-0.940000	-3.880000	10.980000
9	21:48:26.942	a	134465	15.650000	20.860000	23.560000	20.360000	-0.940000	-3.880000	10.980000
10	21:48:26.992	a	134567	15.650000	20.860000	23.560000	20.360000	-0.860000	-3.800000	11.250000
11	21:48:27.043	a	134567	15.650000	20.860000	23.560000	20.360000	-0.860000	-3.800000	11.250000
12	21:48:27.093	a	134669	15.650000	20.860000	23.560000	20.360000	-1.720000	-4.270000	11.370000
13	21:48:27.144	a	134669	15.650000	20.860000	23.560000	20.360000	-1.720000	-4.270000	11.370000
14	21:48:27.194	a	134771	15.650000	20.840000	23.560000	20.360000	-0.980000	-4.110000	10.860000
15	21:48:27.245	a	134771	15.650000	20.840000	23.560000	20.360000	-0.980000	-4.110000	10.860000
16	21:48:27.295	a	134874	15.610000	20.840000	23.560000	20.320000	-1.920000	-3.880000	11.060000
17	21:48:27.346	a	134874	15.610000	20.840000	23.560000	20.320000	-1.920000	-3.880000	11.060000

Cleaned Data

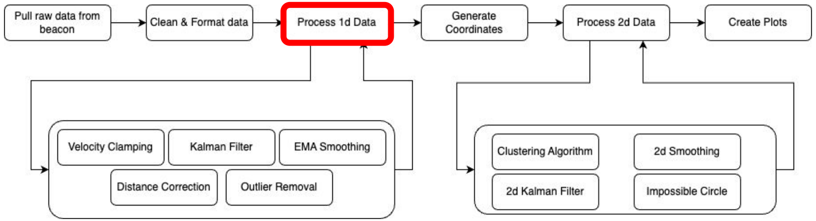


	timestamp	playerid	wearabletimestamp	b1d	b2d	b3d	b4d	xa	ya	za
2	21:48:26.677	a	134159	15.690000	20.800000	23.560000	20.370000	-1.010000	-4.470000	10.940000
3	21:48:26.677	a	134159	15.690000	20.800000	23.560000	20.370000	-1.010000	-4.470000	10.940000
4	21:48:26.691	a	134261	15.690000	20.800000	23.560000	20.370000	-0.620000	-4.430000	10.390000
5	21:48:26.741	a	134261	15.690000	20.800000	23.560000	20.370000	-0.620000	-4.430000	10.390000
6	21:48:26.791	a	134363	15.580000	20.830000	23.560000	20.400000	-0.540000	-4.430000	10.900000
7	21:48:26.842	a	134363	15.580000	20.830000	23.560000	20.400000	-0.540000	-4.430000	10.900000
8	21:48:26.892	a	134465	15.650000	20.860000	23.560000	20.360000	-0.940000	-3.880000	10.980000
9	21:48:26.942	a	134465	15.650000	20.860000	23.560000	20.360000	-0.940000	-3.880000	10.980000
10	21:48:26.992	a	134567	15.650000	20.860000	23.560000	20.360000	-0.860000	-3.800000	11.250000
11	21:48:27.043	a	134567	15.650000	20.860000	23.560000	20.360000	-0.860000	-3.800000	11.250000
12	21:48:27.093	a	134669	15.650000	20.860000	23.560000	20.360000	-1.720000	-4.270000	11.370000
13	21:48:27.144	a	134669	15.650000	20.860000	23.560000	20.360000	-1.720000	-4.270000	11.370000
14	21:48:27.194	a	134771	15.650000	20.840000	23.560000	20.360000	-0.980000	-4.110000	10.860000
15	21:48:27.245	a	134771	15.650000	20.840000	23.560000	20.360000	-0.980000	-4.110000	10.860000
16	21:48:27.295	a	134874	15.610000	20.840000	23.560000	20.320000	-1.920000	-3.880000	11.060000
17	21:48:27.346	a	134874	15.610000	20.840000	23.560000	20.320000	-1.920000	-3.880000	11.060000

Cleaned Data

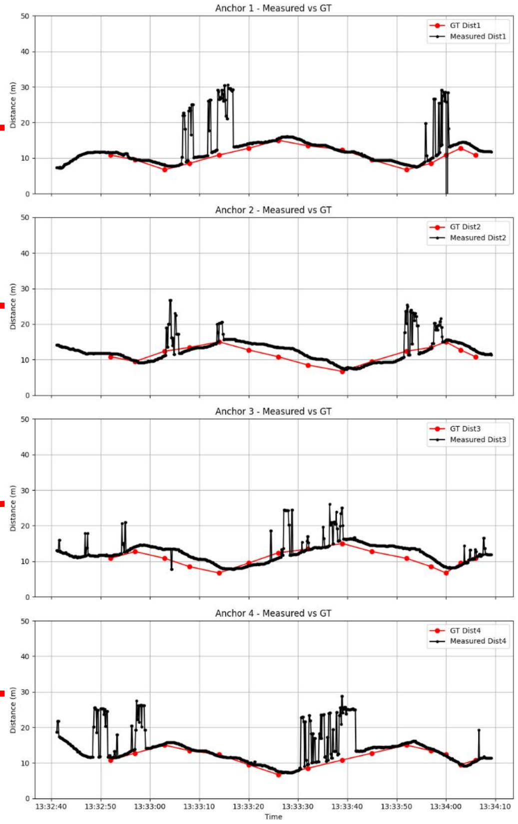


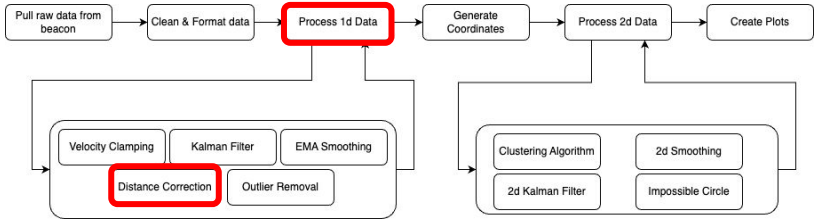
	timestamp	playerid	wearabletimestamp	b1d	b2d	b3d	b4d	xa	ya	za
2	21:48:26.677	a	134159	15.690000	20.800000	23.560000	20.370000	-1.010000	-4.470000	10.940000
3	21:48:26.677	a	134159	15.690000	20.800000	23.560000	20.370000	-1.010000	-4.470000	10.940000
4	21:48:26.691	a	134261	15.690000	20.800000	23.560000	20.370000	-0.620000	-4.430000	10.390000
5	21:48:26.741	a	134261	15.690000	20.800000	23.560000	20.370000	-0.620000	-4.430000	10.390000
6	21:48:26.791	a	134363	15.580000	20.830000	23.560000	20.400000	-0.540000	-4.430000	10.900000
7	21:48:26.842	a	134363	15.580000	20.830000	23.560000	20.400000	-0.540000	-4.430000	10.900000
8	21:48:26.892	a	134465	15.650000	20.860000	23.560000	20.360000	-0.940000	-3.880000	10.980000
9	21:48:26.942	a	134465	15.650000	20.860000	23.560000	20.360000	-0.940000	-3.880000	10.980000
10	21:48:26.992	a	134567	15.650000	20.860000	23.560000	20.360000	-0.860000	-3.800000	11.250000
11	21:48:27.043	a	134567	15.650000	20.860000	23.560000	20.360000	-0.860000	-3.800000	11.250000
12	21:48:27.093	a	134669	15.650000	20.860000	23.560000	20.360000	-1.720000	-4.270000	11.370000
13	21:48:27.144	a	134669	15.650000	20.860000	23.560000	20.360000	-1.720000	-4.270000	11.370000
14	21:48:27.194	a	134771	15.650000	20.840000	23.560000	20.360000	-0.980000	-4.110000	10.860000
15	21:48:27.245	a	134771	15.650000	20.840000	23.560000	20.360000	-0.980000	-4.110000	10.860000
16	21:48:27.295	a	134874	15.610000	20.840000	23.560000	20.320000	-1.920000	-3.880000	11.060000
17	21:48:27.346	a	134874	15.610000	20.840000	23.560000	20.320000	-1.920000	-3.880000	11.060000



1d Processing

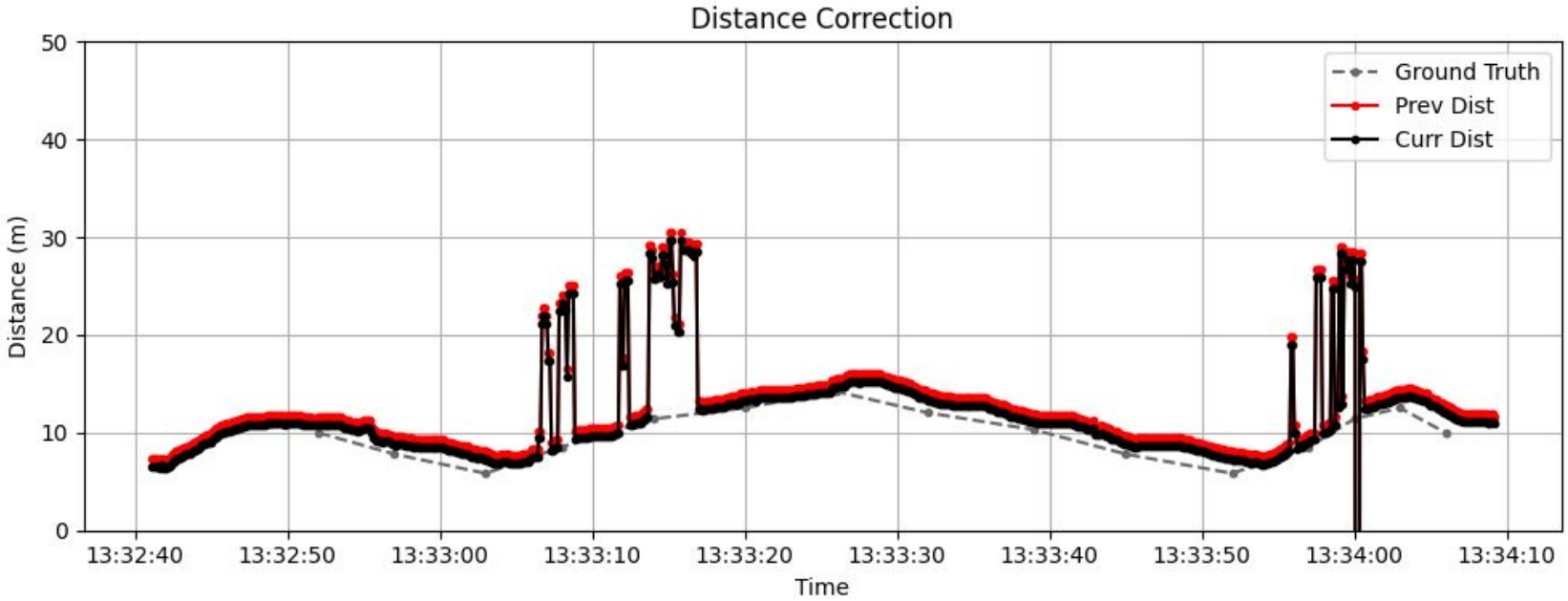
```
timestamp,playerid,wea       estamp, b1d,b2d,b3d,b4d,xs,ys,zs
21:48:26.677,a,134159,5.690000,0.800000,3.560000,0.370000,-1.010000,-4.470000,10.940000
21:48:26.677,a,134159,5.690000,0.800000,3.560000,0.370000,-1.010000,-4.470000,10.940000
21:48:26.691,a,134261,5.690000,0.800000,3.560000,0.370000,-0.620000,-4.430000,10.390000
21:48:26.741,a,134261,5.690000,0.800000,3.560000,0.370000,-0.620000,-4.430000,10.390000
21:48:26.791,a,134363,5.580000,0.830000,3.560000,0.400000,-0.540000,-4.430000,10.900000
21:48:26.842,a,134363,5.580000,0.830000,3.560000,0.400000,-0.540000,-4.430000,10.900000
21:48:26.892,a,134465,5.650000,0.860000,3.560000,0.360000,-0.940000,-3.880000,10.980000
21:48:26.942,a,134465,5.650000,0.860000,3.560000,0.360000,-0.940000,-3.880000,10.980000
21:48:26.992,a,134567,5.650000,0.860000,3.560000,0.360000,-0.860000,-3.800000,11.250000
21:48:27.043,a,134567,5.650000,0.860000,3.560000,0.360000,-0.860000,-3.800000,11.250000
21:48:27.093,a,134669,5.650000,0.860000,3.560000,0.360000,-1.720000,-4.270000,11.370000
21:48:27.144,a,134669,5.650000,0.860000,3.560000,0.360000,-1.720000,-4.270000,11.370000
21:48:27.194,a,134771,5.650000,0.840000,3.560000,0.360000,-0.980000,-4.110000,10.860000
21:48:27.245,a,134771,5.650000,0.840000,3.560000,0.360000,-0.980000,-4.110000,10.860000
21:48:27.295,a,134874,5.610000,0.840000,3.560000,0.320000,-1.920000,-3.880000,11.060000
21:48:27.346,a,134874,5.610000,0.840000,3.560000,0.320000,-1.920000,-3.880000,11.060000
```





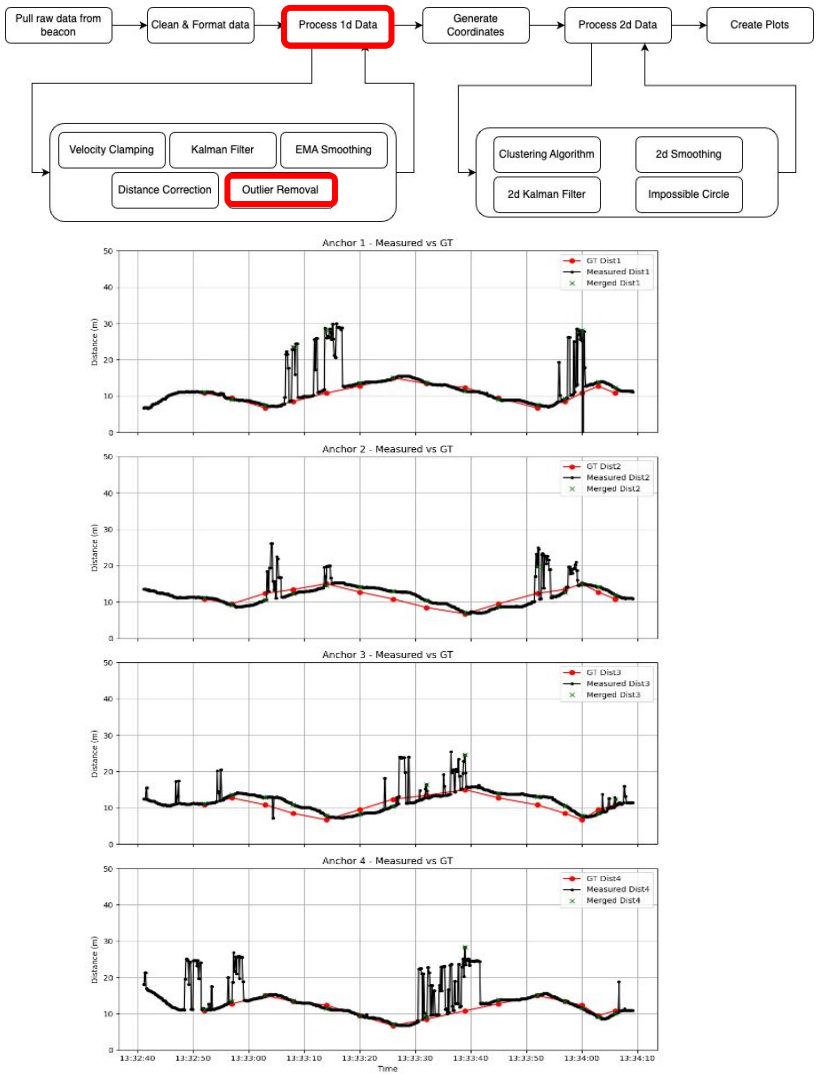
Distance Correction

Systematic overestimate of 0.8 meters is subtracted



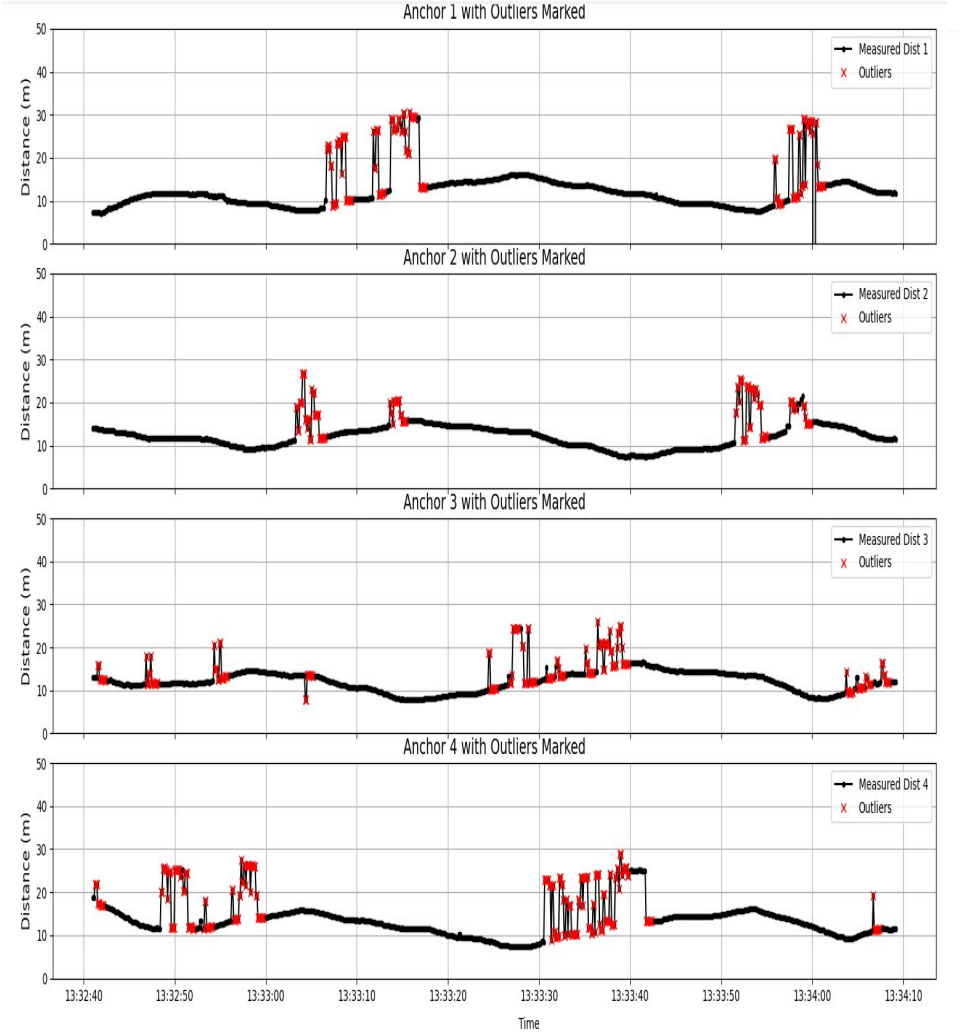
Outlier Removal

- Default window of 800ms
- Default residual variance threshold of 0.5 meters
- Conversion of timestamp column to datetime format



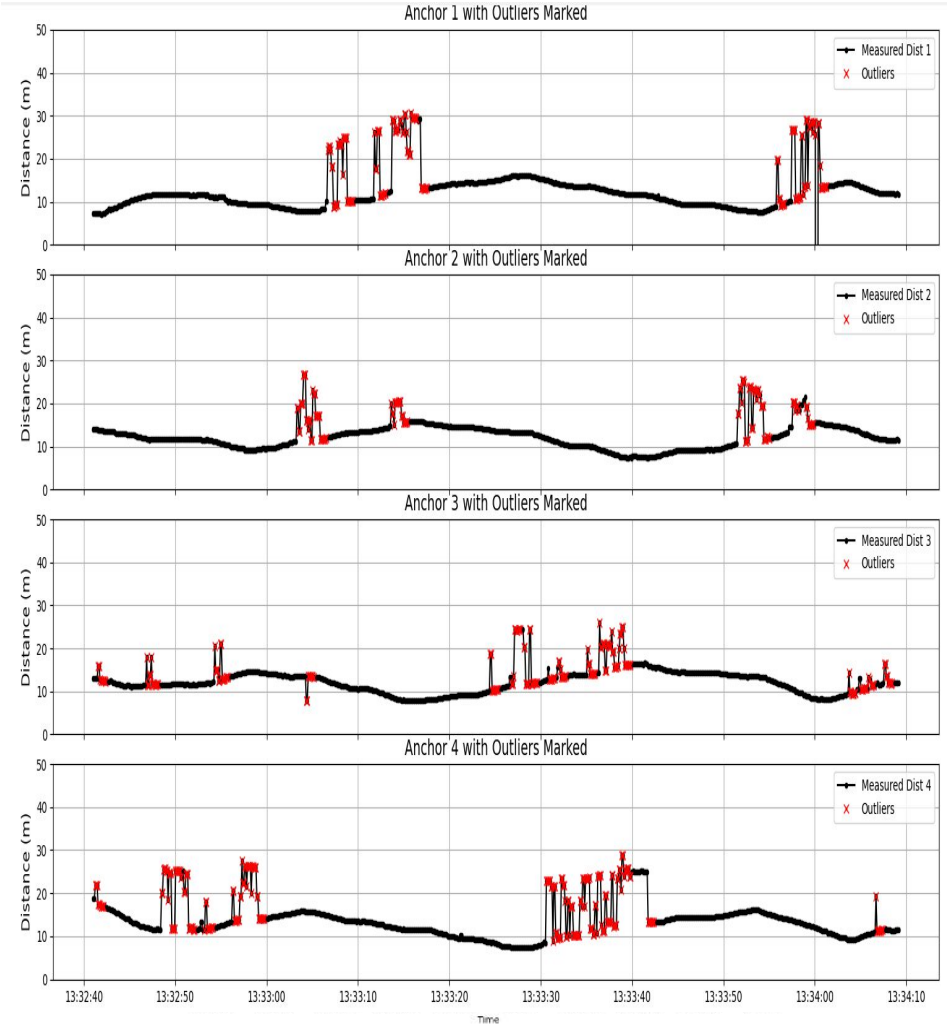
Outlier Removal

- Apply rolling window of 800ms
- Utilize linear regression to a create fitted line
- If residual variance deviates by more than ½ meter, considered outlier

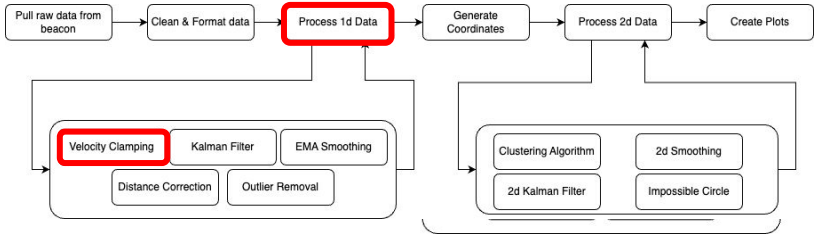


Outlier Removal

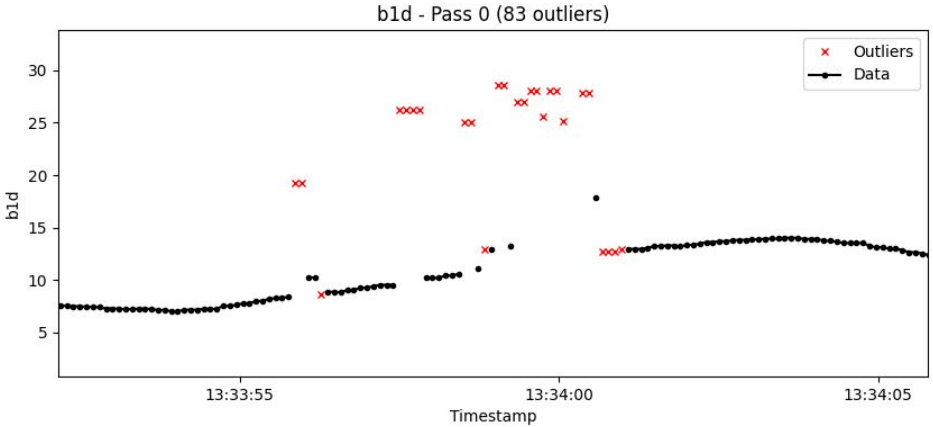
- First Filter: Removes large spikes by comparing data point differences to the median
- Second Filter: Excludes outliers only in the positive direction
- NaN values are filled for outlier points.
- If too few inlier points remain, original fitted line is kept



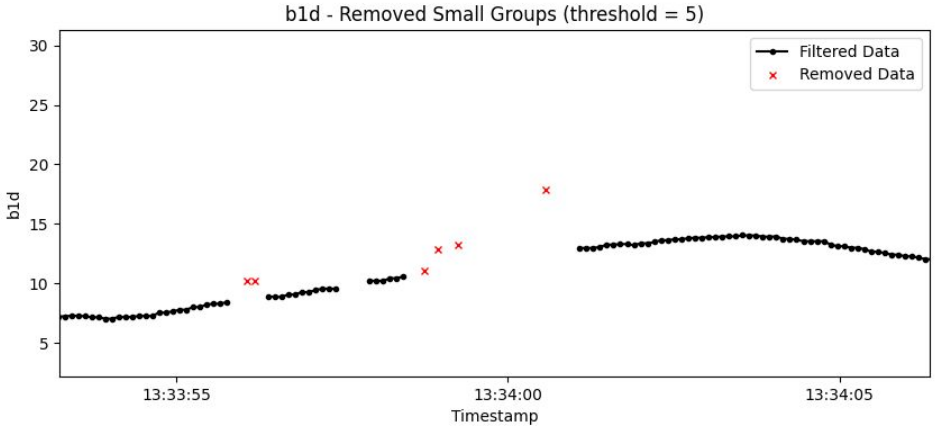
Velocity Clamping



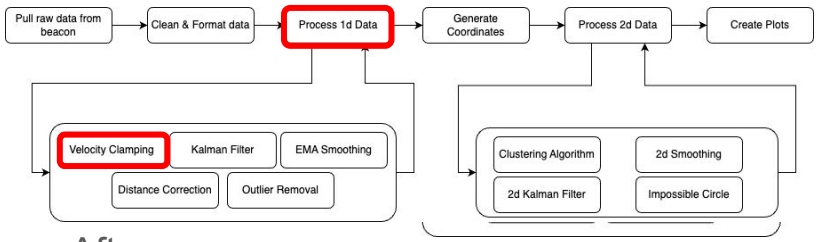
Repeatedly mark points and delete points with too high velocity



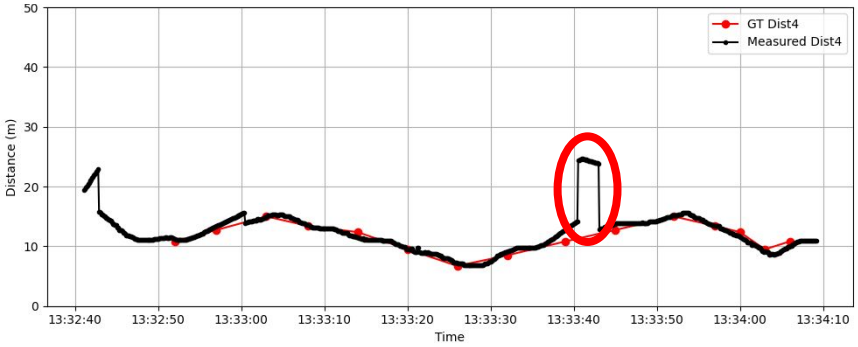
Small floating groups of points are removed

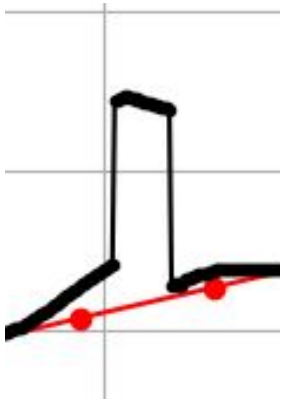


Velocity Clamping

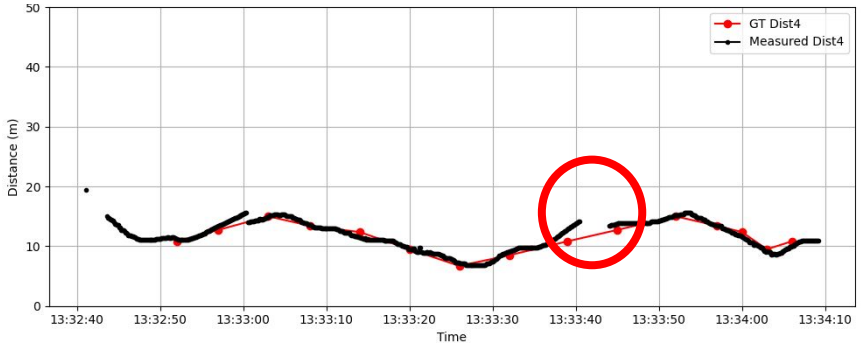


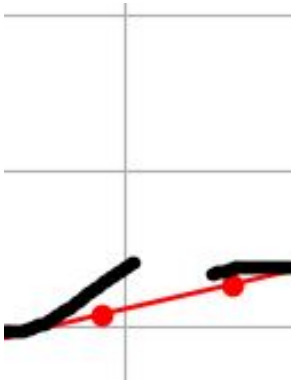
Before



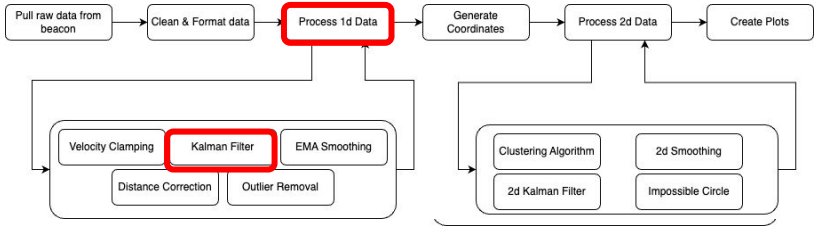


After



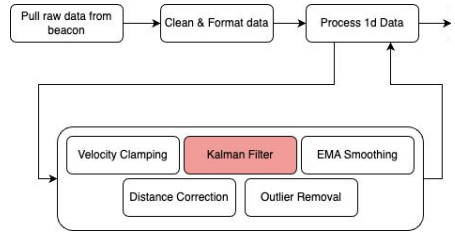


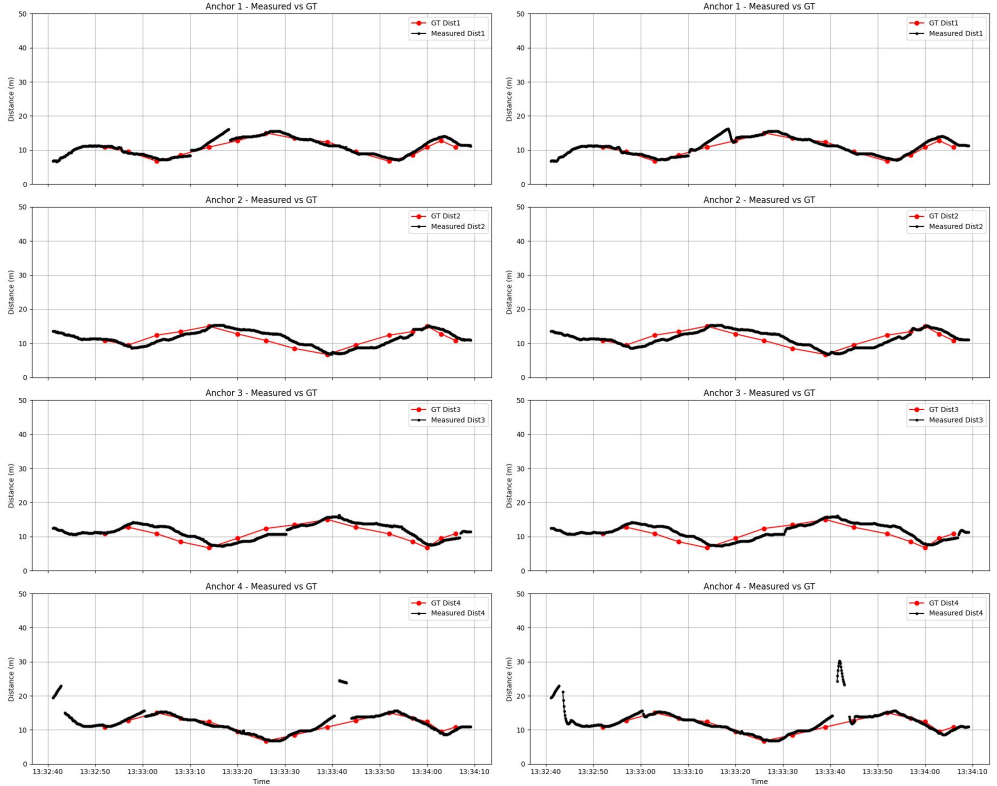
Kalman filtering

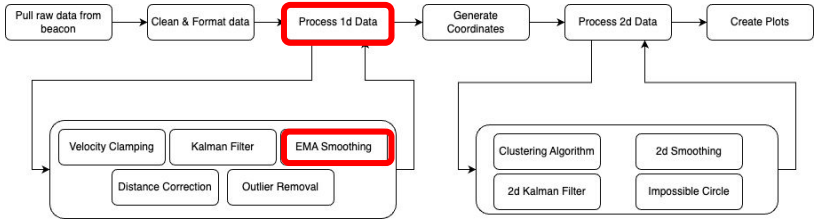


- What is a Kalman filter?
  - Feed it a model, measurements and uncertainties
  - It predicts new value based on model, then takes weighted average between prediction and measurements
  - Should I include matrix eq? Or easier version of it?
- How have we adapted it?
  - Newtonian model w/ constant acceleration(approximation)
  - Nans: use prediction for model, then overwrite with nan
  - Use innovation (measurement - prediction) and residual variance to inflate uncertainty in the measurement

What does Kalman filtering look like?



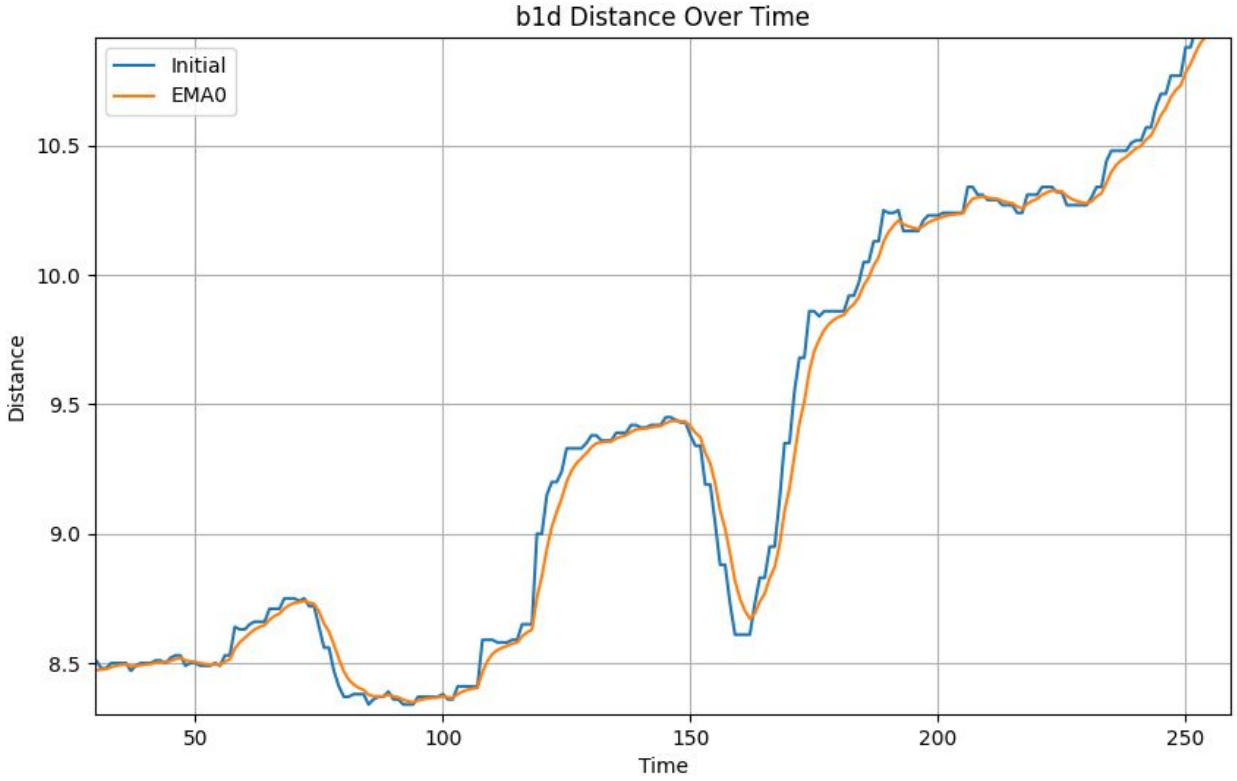




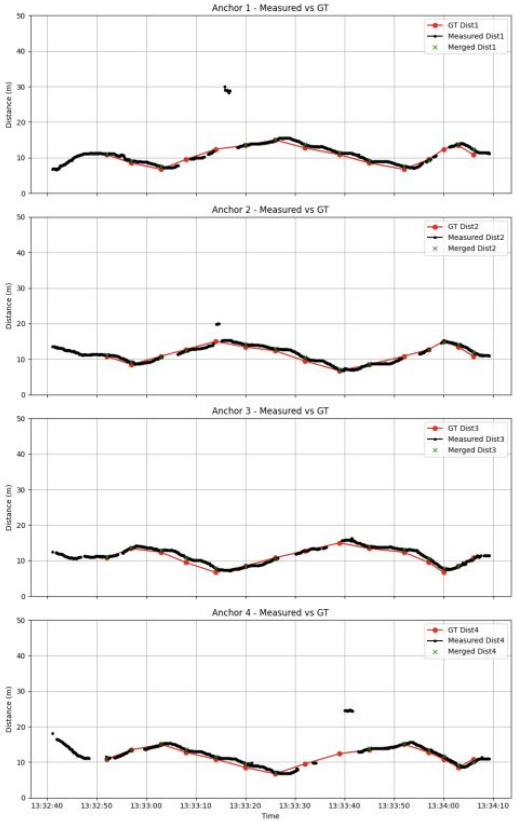
Exponential Moving Average

Smoothes data

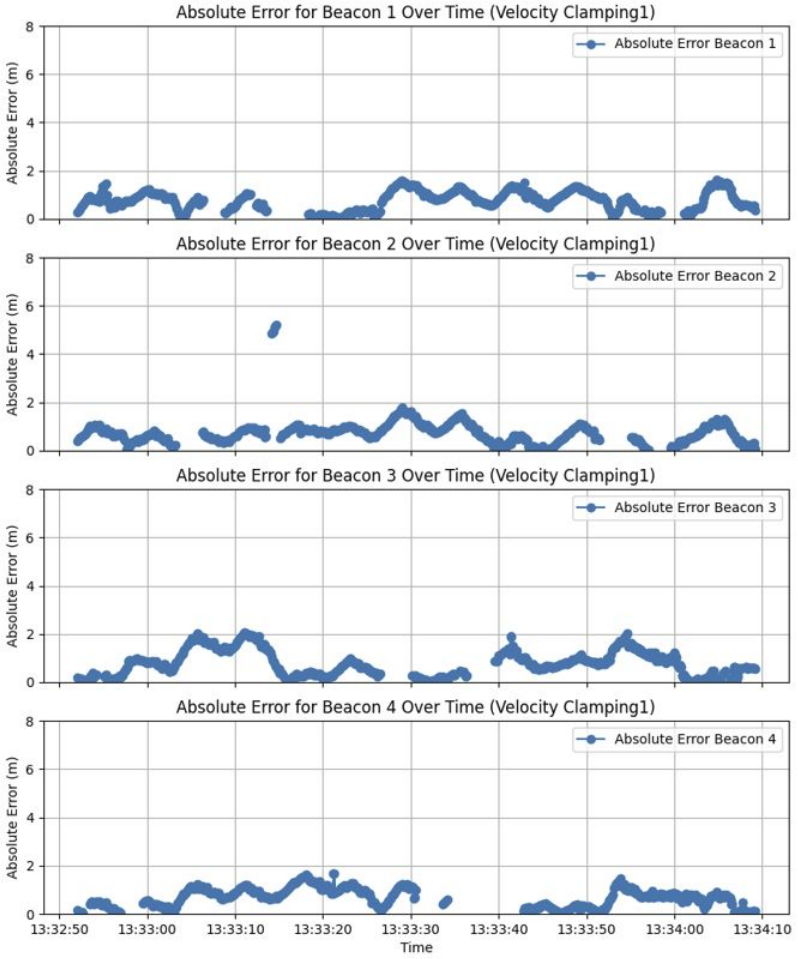
Take a window then calculates the average for that window.



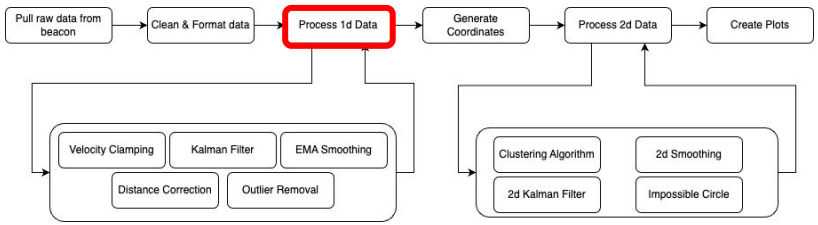
Mean Absolute Error (MAE)

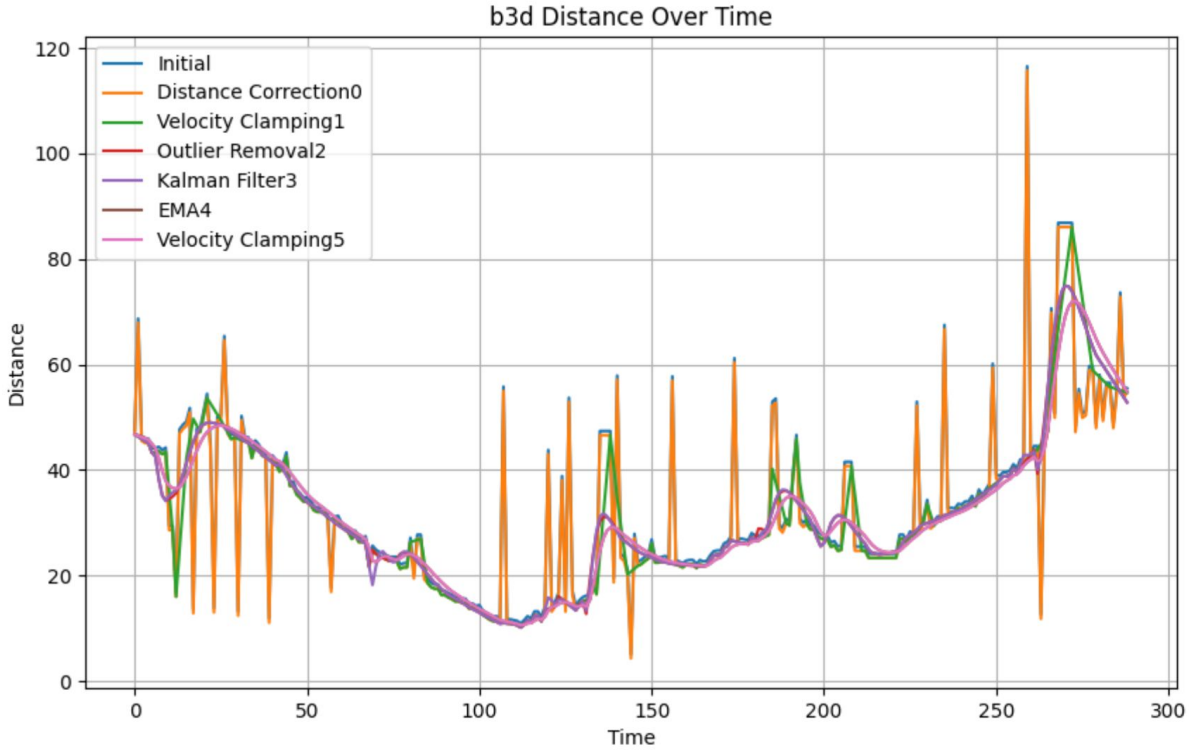


Beacon 1: MAE=0.590 m
Beacon 2: MAE=0.512 m
Beacon 3: MAE=0.671 m
Beacon 4: MAE=0.518 m

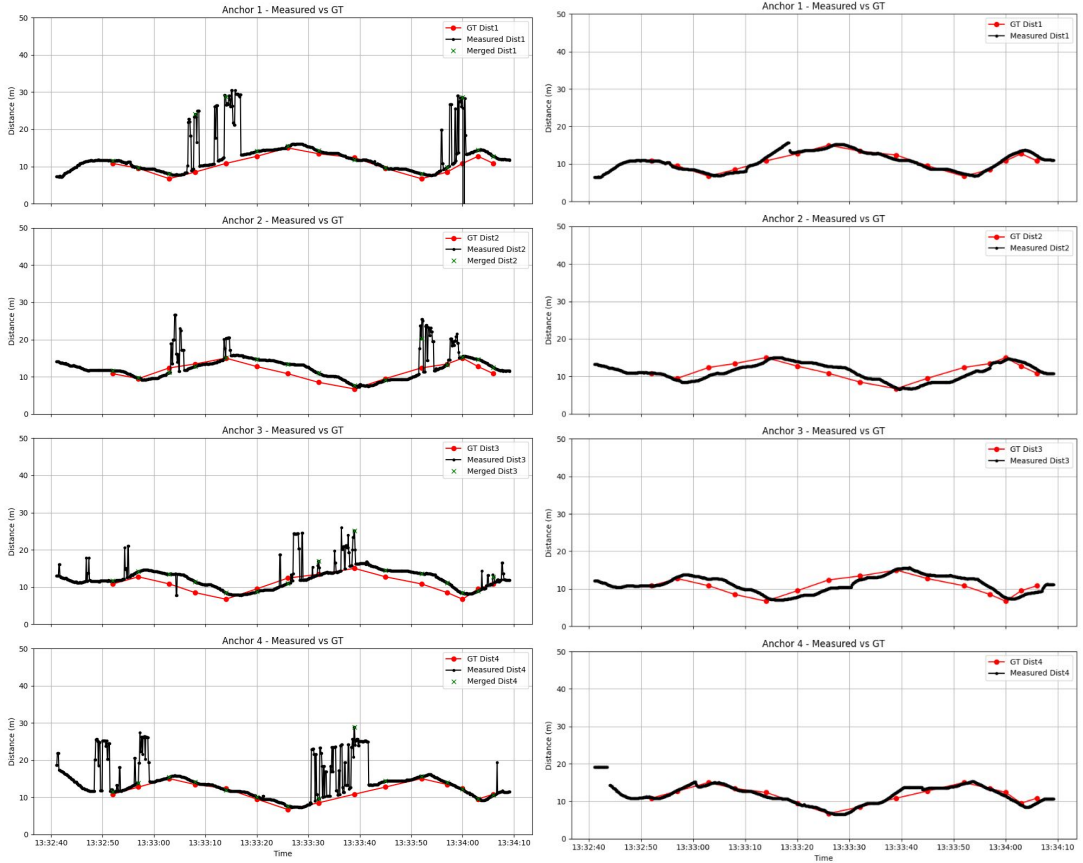


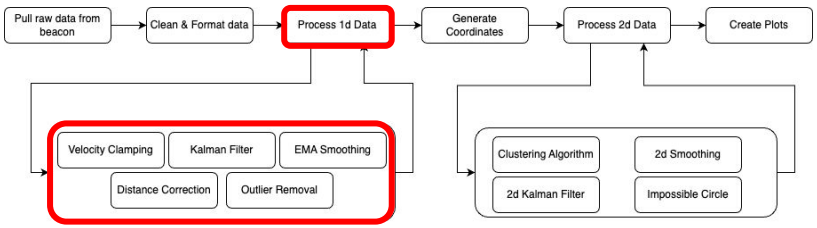
Raw Data vs. Processed Data





Processing Results



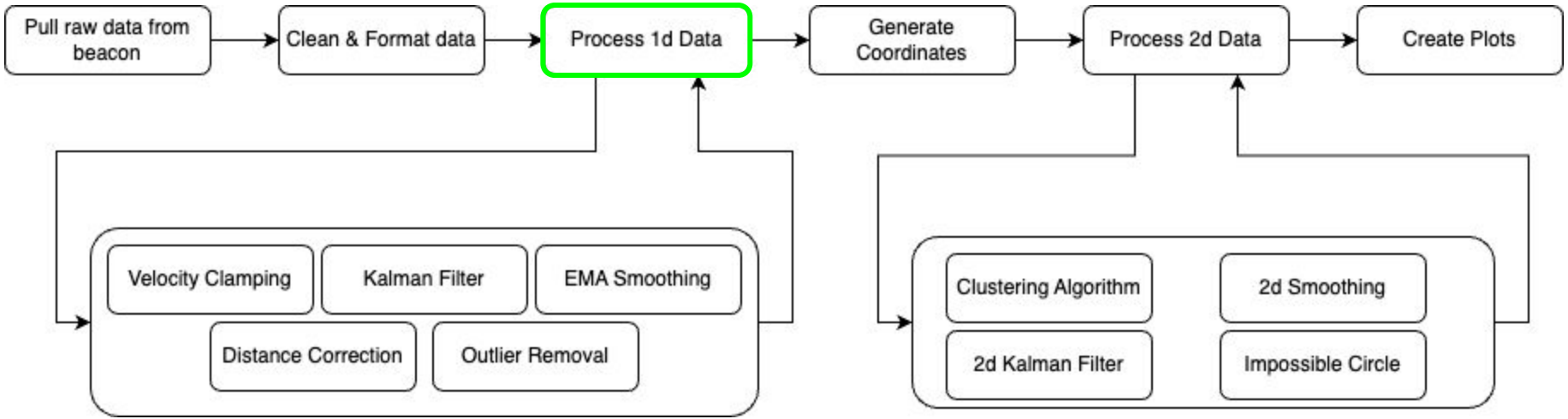


Left: Raw Data
Right: Processed Data

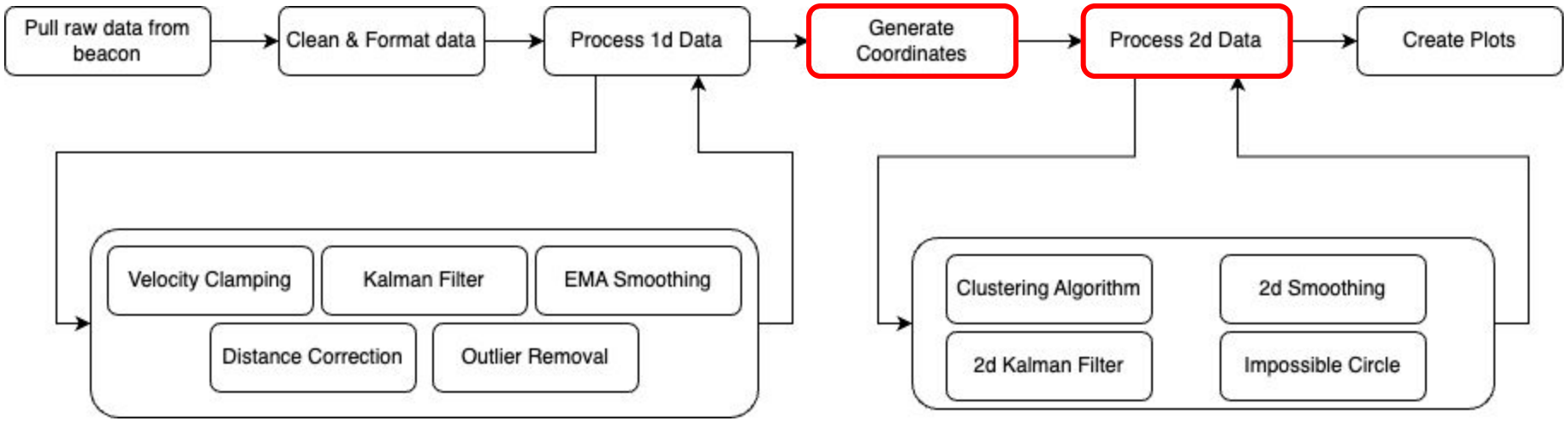
Note the removal of the large spikes from the left to the right.

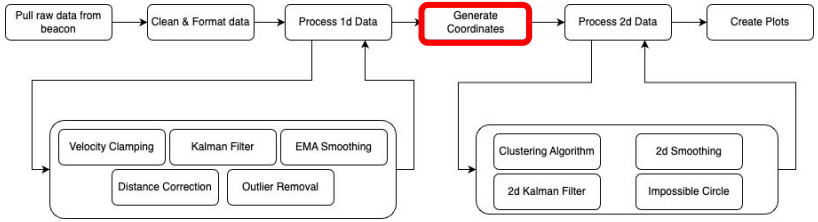
Raw MAE: 2-4m
Processed MAE: 0.5-1.3m

1d Processing Complete!



2d Processing

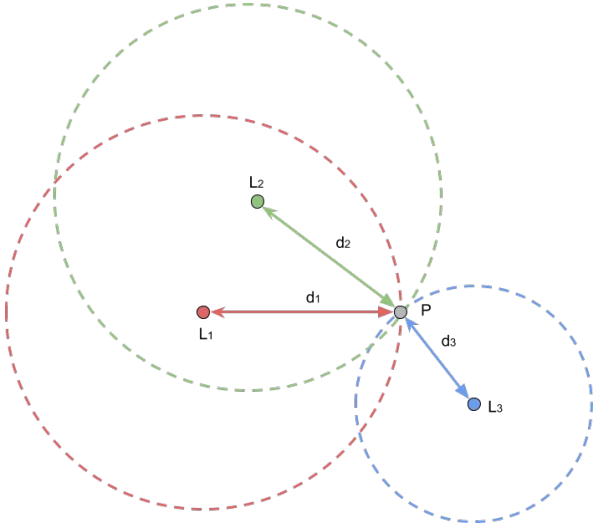


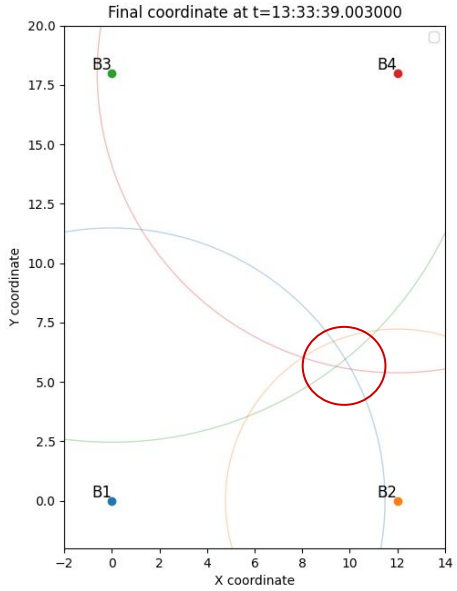


Determine Position Coordinates

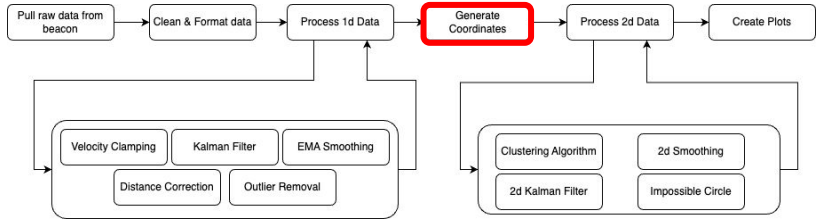
Common strategy: trilateration

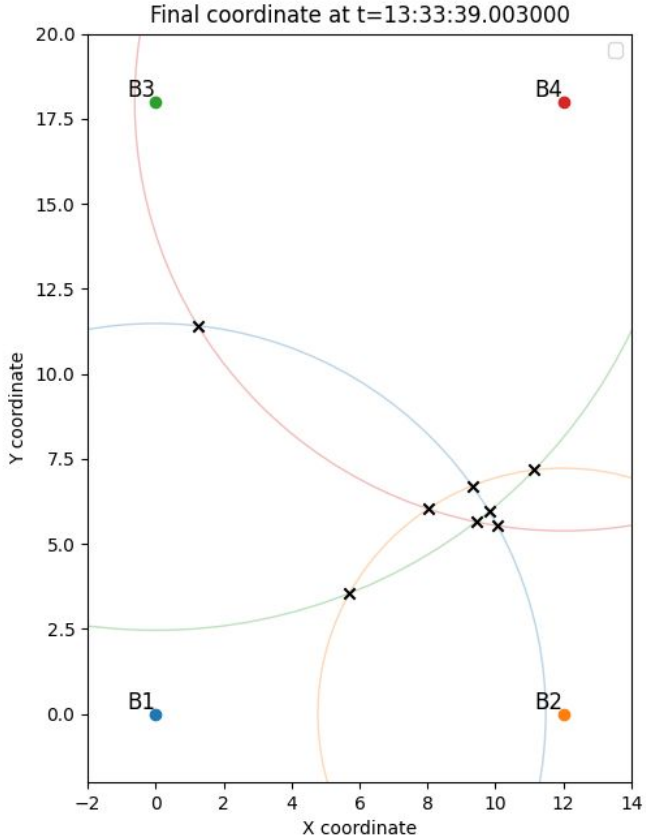
What do we do when the input data is not perfect?



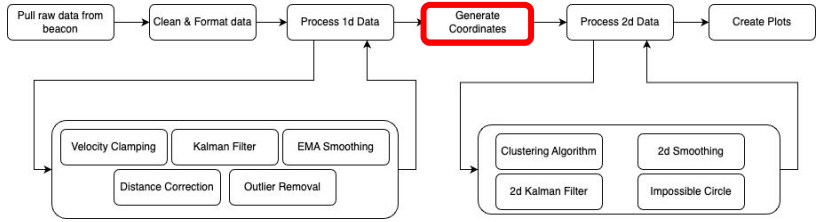


Determine Position Coordinates

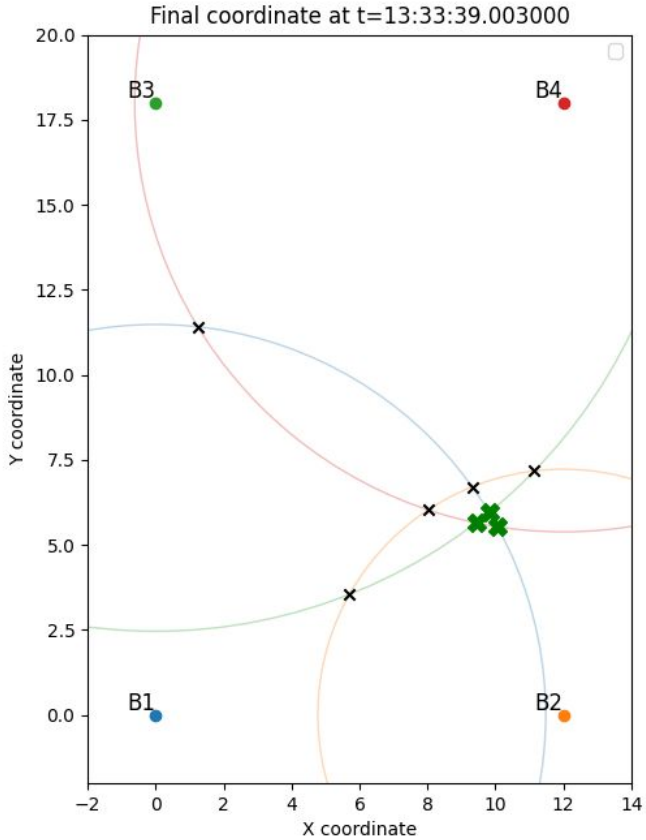




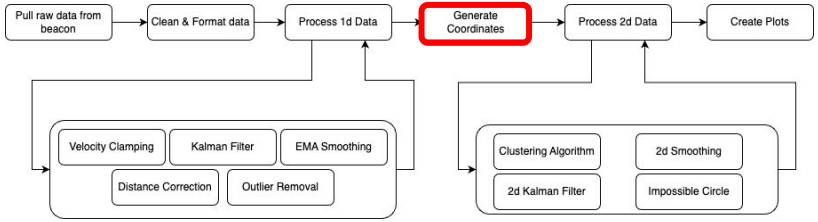
- Draw circles with the measured radii
- Find all intersections
- Find the densest cluster
- Find the centroid - this is the position coordinate
- Determine confidence value by measuring the distance to the circles



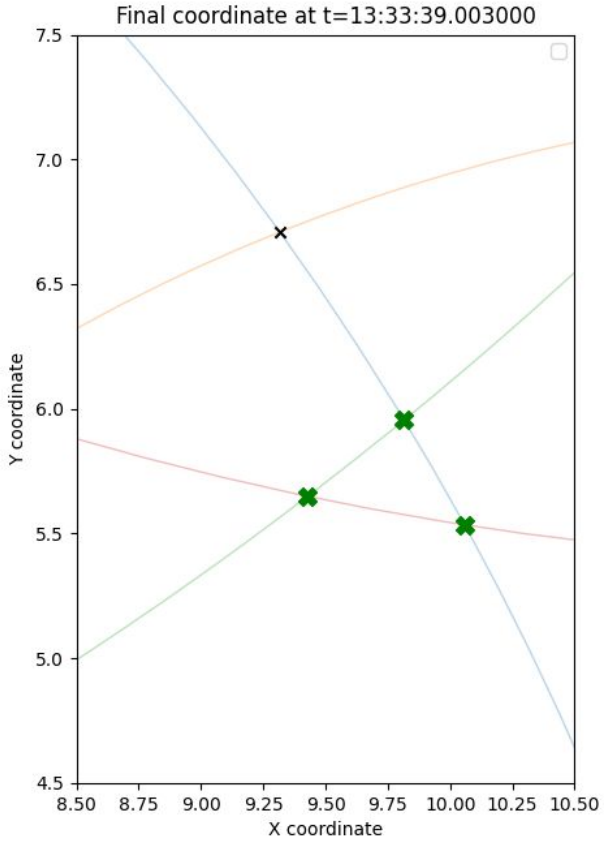
Determine Position Coordinates



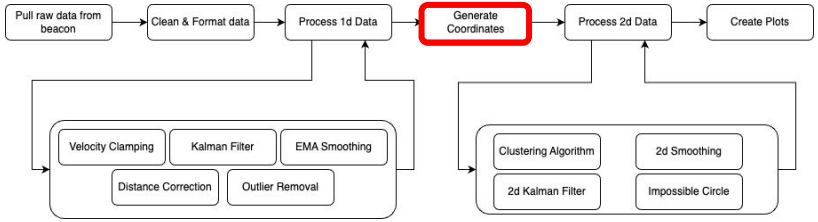
- Draw circles with the measured radii
- Find all intersections
- Find the densest cluster
- Find the centroid - this is the position coordinate
- Determine confidence value by measuring the distance to the circles



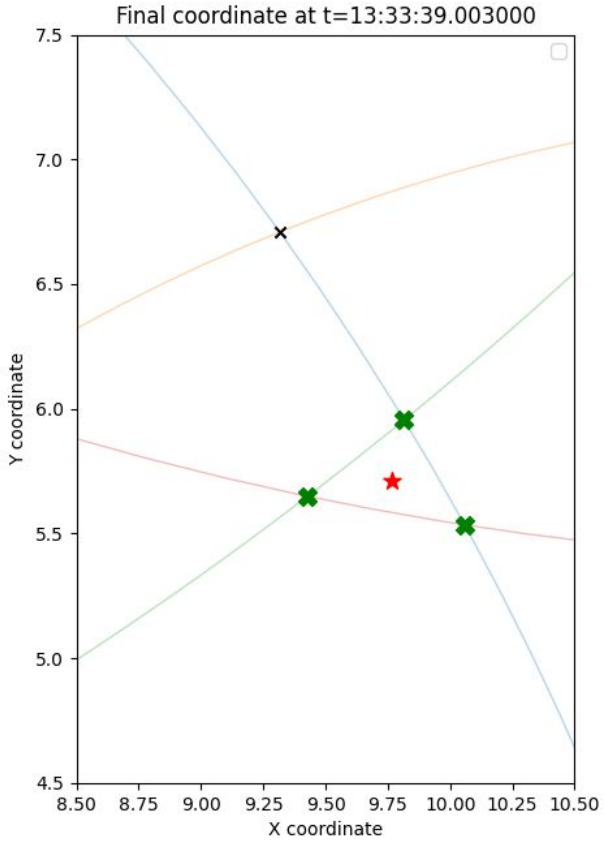
Determine Position Coordinates



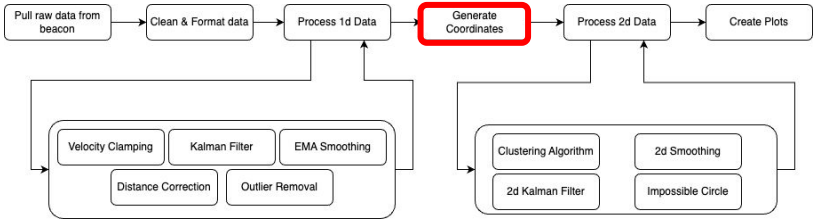
- Draw circles with the measured radii
- Find all intersections
- Find the densest cluster
- Find the centroid - this is the position coordinate
- Determine confidence value by measuring the distance to the circles



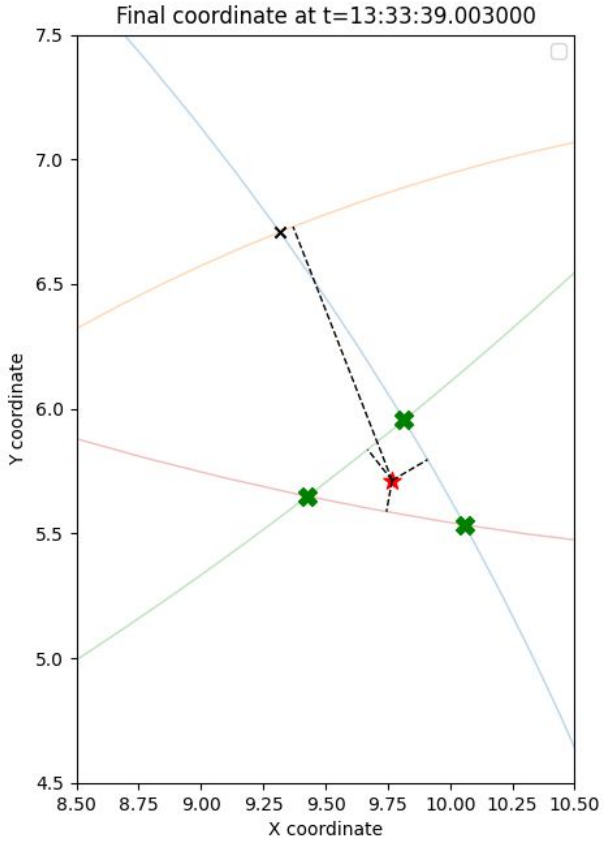
Determine Position Coordinates



- Draw circles with the measured radii
- Find all intersections
- Find the densest cluster
- Find the centroid - this is the position coordinate
- Determine confidence value by measuring the distance to the circles

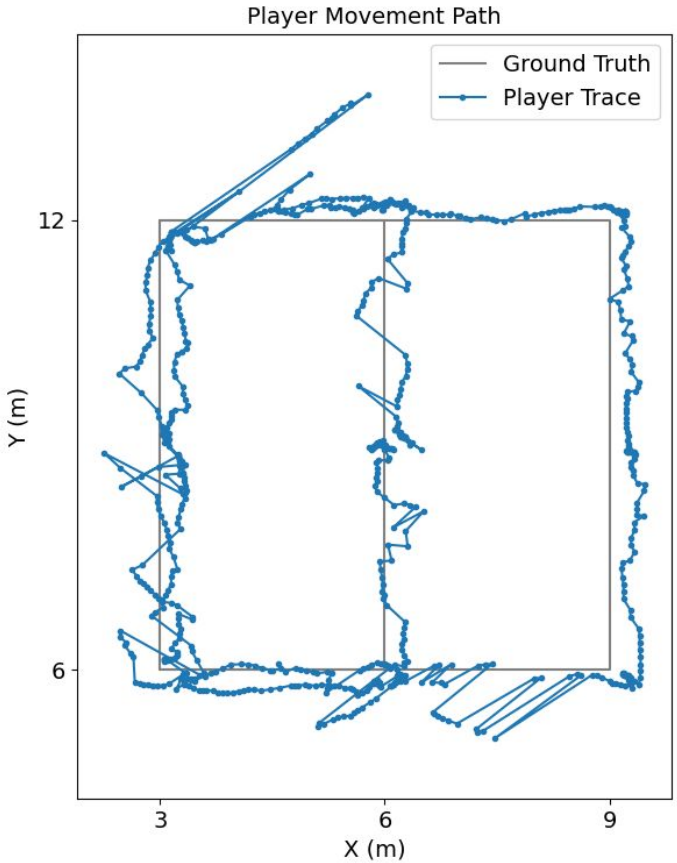


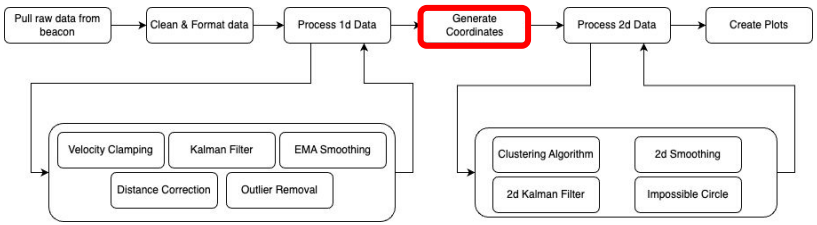
Determine Position Coordinates



- Draw circles with the measured radii
- Find all intersections
- Find the densest cluster
- Find the centroid - this is the position coordinate
- Determine confidence value by measuring the distance to the circles

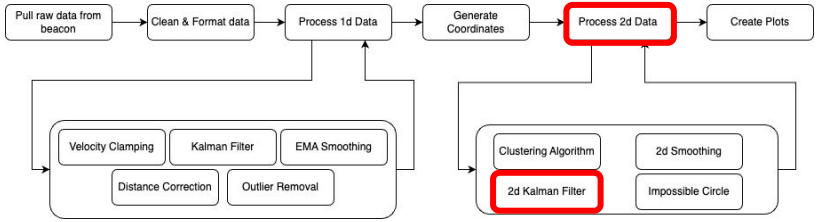
Determine Position Coordinates





…a little messy…

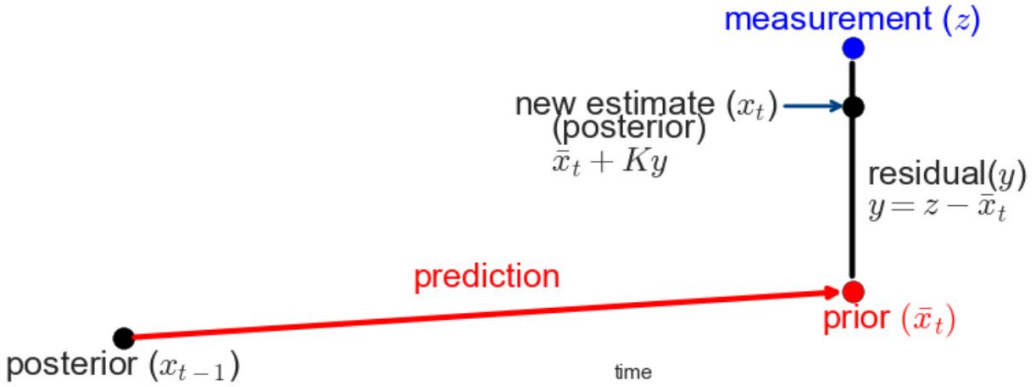
Can we do better?

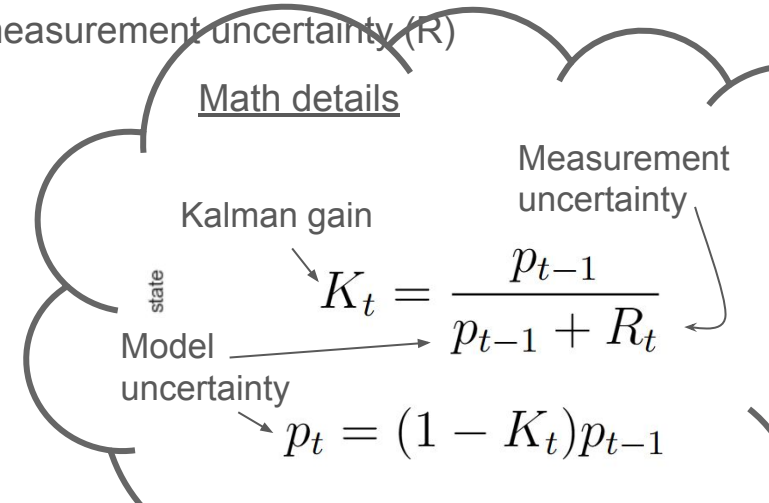


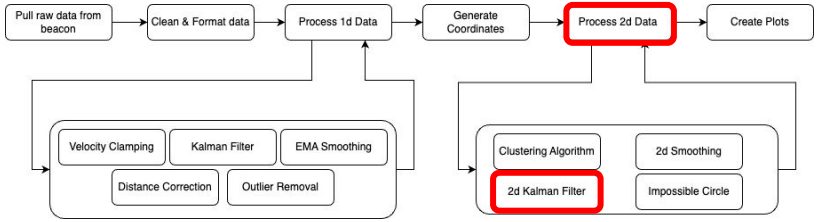
Kalman Filtering

- What is a Kalman filter?
  - Input a model, measurements and uncertainties
  - Predicts new estimate based on model, then takes weighted average between prediction and measurements and updates model uncertainties

- How have we adapted it?
  - Kinematic model with constant acceleration
  - Set an upper limit on change in position based on maximum velocity and measured acceleration
  - Use confidence value to dynamically inflate measurement uncertainty (R)







Kalman Filtering

What is a Kalman filter?

- Cycles of predictions, measurements and final estimates
- Input a model, measurements (z) and uncertainties (R)
- Uses *Kalman gain* (K) to find a weighted average between model prediction($\overline{x}_t$) and measurement (z) to find new estimate (x_t)
- Updates model uncertainty (p)

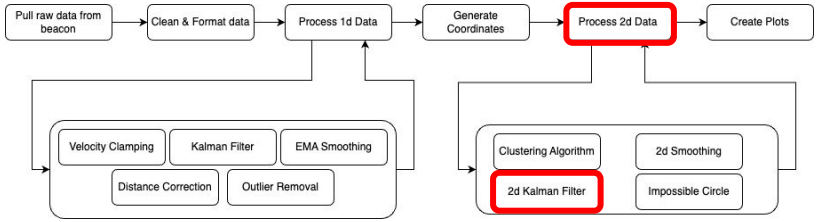
Math details

$$x_t = x_{t-1} + v\Delta t + \frac{1}{2}a(\Delta t)^2$$

$$K_t = \frac{p_{t-1}}{p_{t-1} + R_t}$$

$$x_t = \overline{x}_t + K_t\ y$$

$$p_t = (1 - K_t)p_{t-1}$$



Kalman Filtering

What is a Kalman filter?

- Cycles of predictions, measurements and final estimates
- Uses kinematic model to predict next step ($\overline{x}_t$)
- Input measurement (z) and uncertainty (R)
- Finds residual (y) between measurement and prediction
- Uses *Kalman gain* (K) and residual to find new estimate (x_t)
- Updates model uncertainty (p)

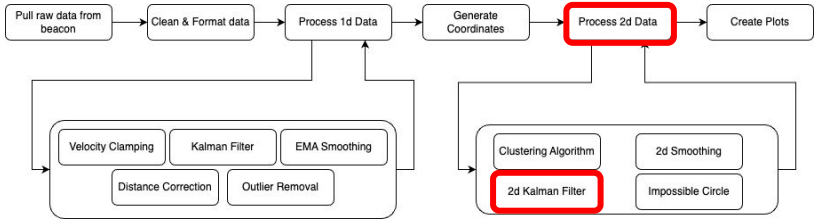
Math details

$$\overline{x}_t = x_{t-1} + v\Delta t + \frac{1}{2}a(\Delta t)^2$$

$$K_t = \frac{p_{t-1}}{p_{t-1} + R_t}$$

$$x_t = \overline{x}_t + K_t\ y$$

$$p_t = (1 - K_t)p_{t-1}$$

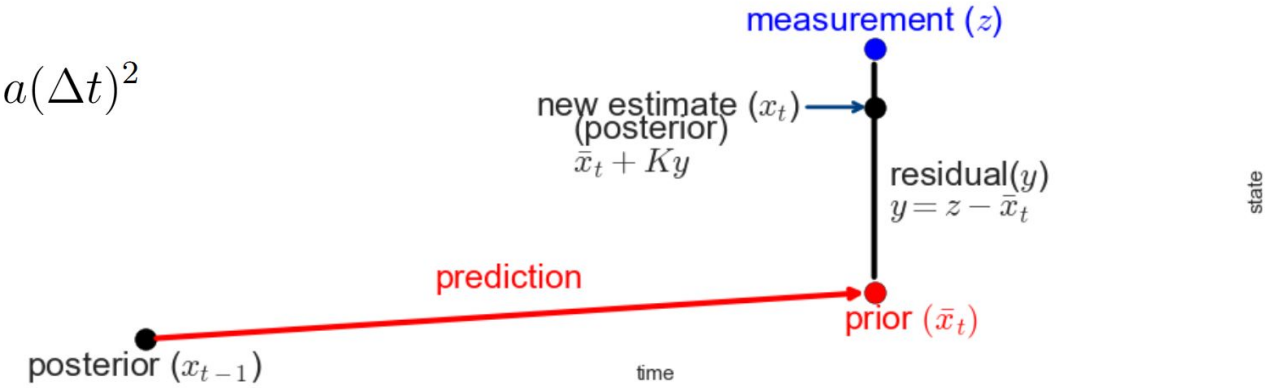


Kalman Filtering

What is a Kalman filter?

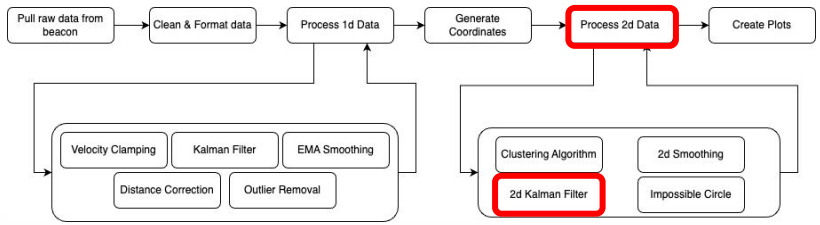
- Cycles of predictions, measurements and final estimates
- Input a model, measurements and uncertainties
- Predicts new estimate based on model, then takes weighted average between prediction and measurements and updates model uncertainties

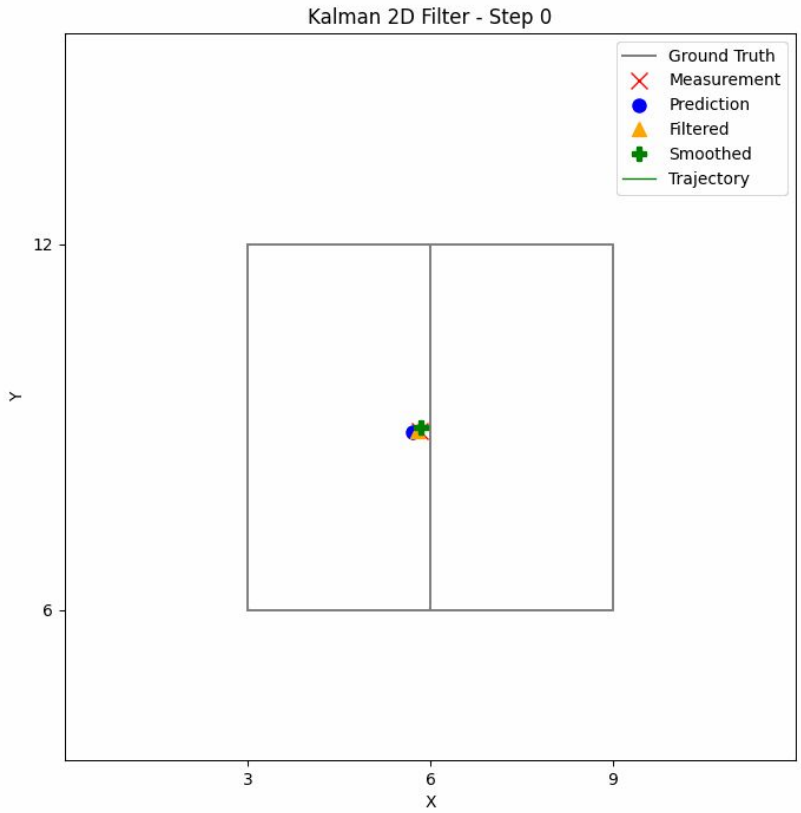
$$x_t = x_{t-1} + v\Delta t + \frac{1}{2}a(\Delta t)^2$$



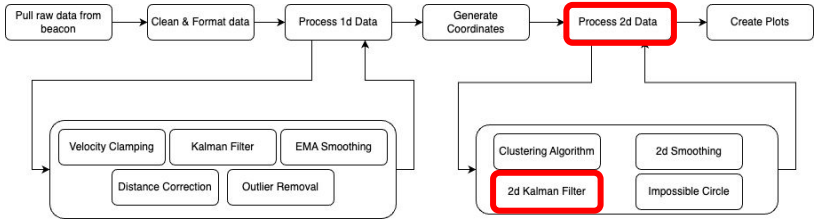
Kalman filtering the 2D trace

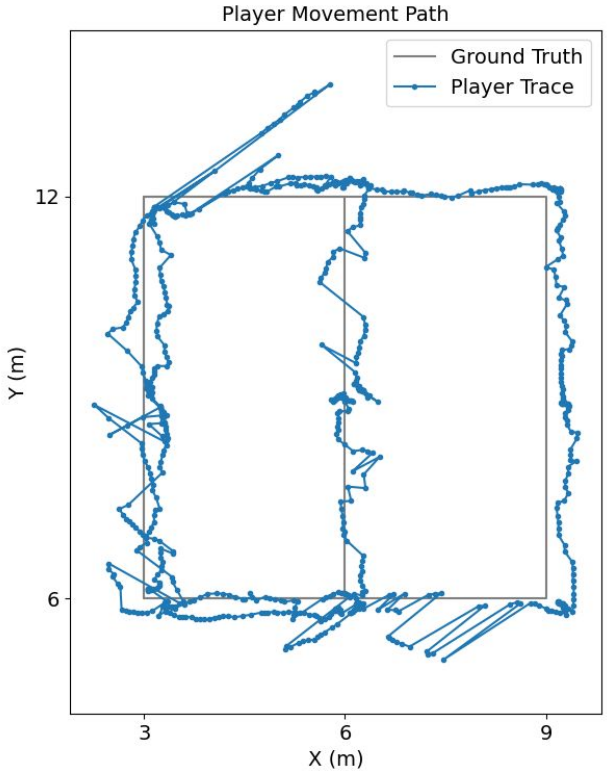
- Input coordinate and confidence to Kalman filter
- Use confidence to dynamically adapt measurement uncertainty (R)
- Apply velocity clamping
- Apply RTS smoothing
  - “Backwards” Kalman filter
  - Moves backwards in time over input to balance out forward bias



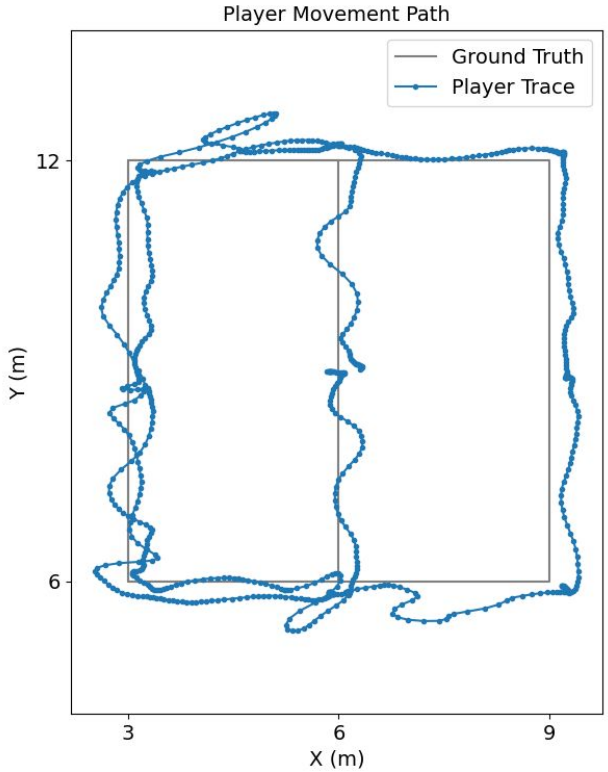


Final Result



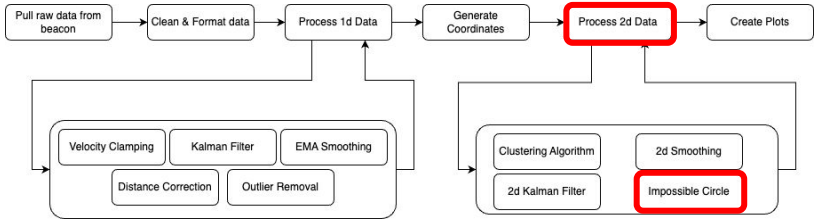


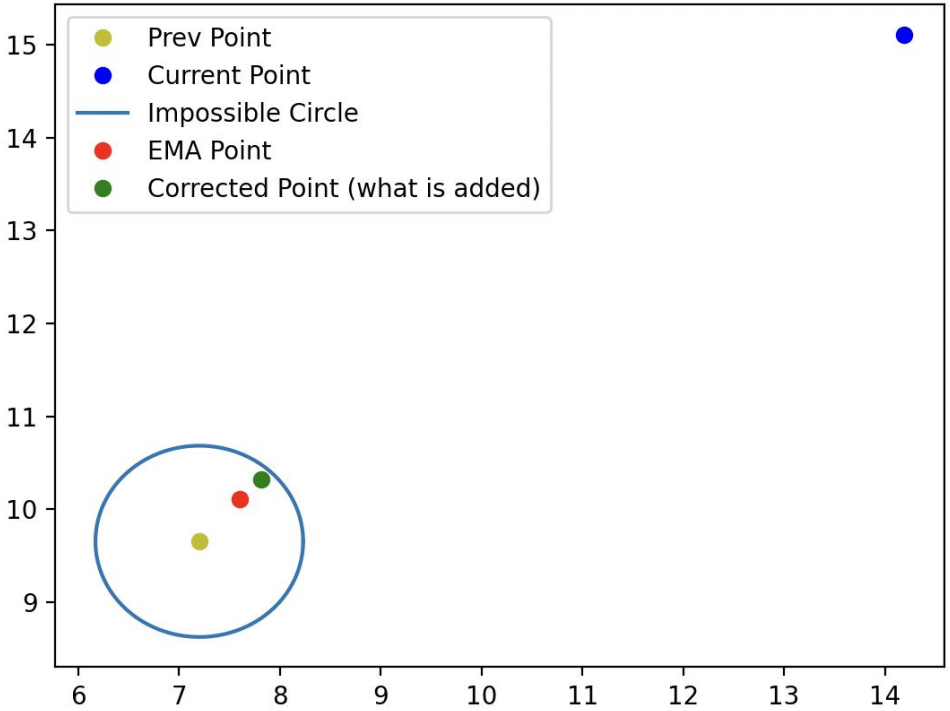
Before Kalman



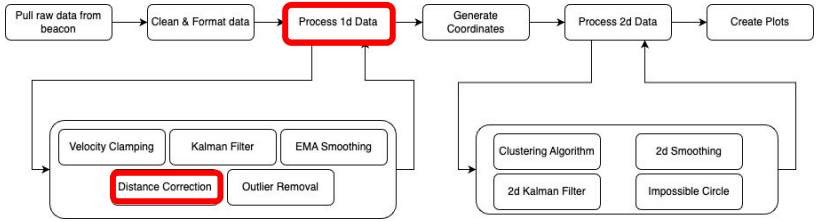
After Kalman

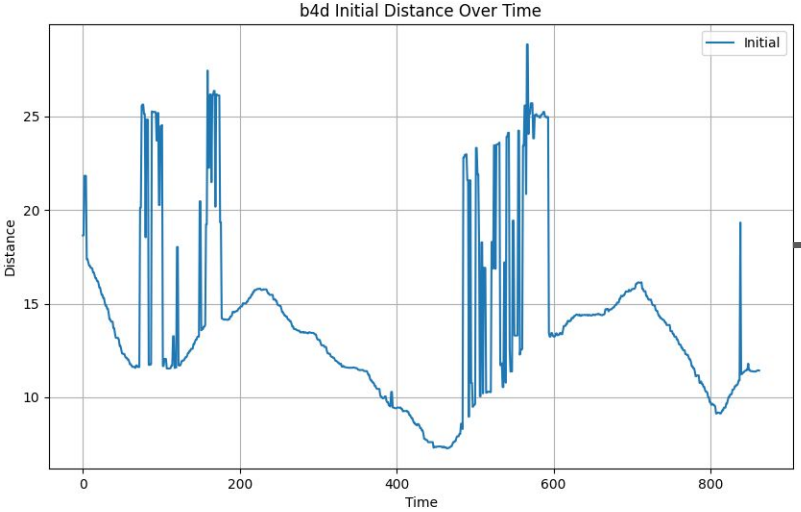
“Impossible Circle”

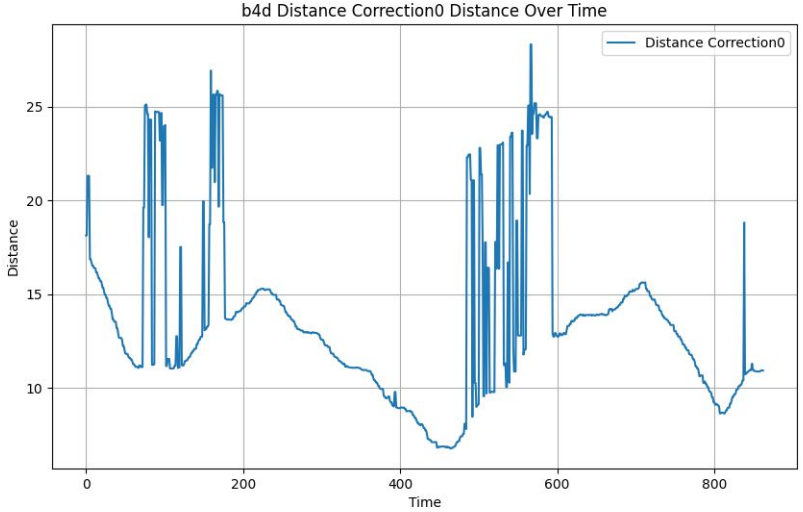




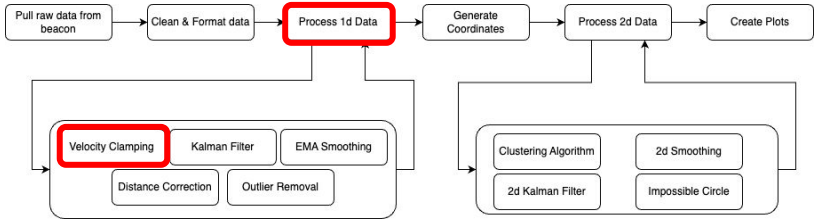
Let’s Process Some Data!

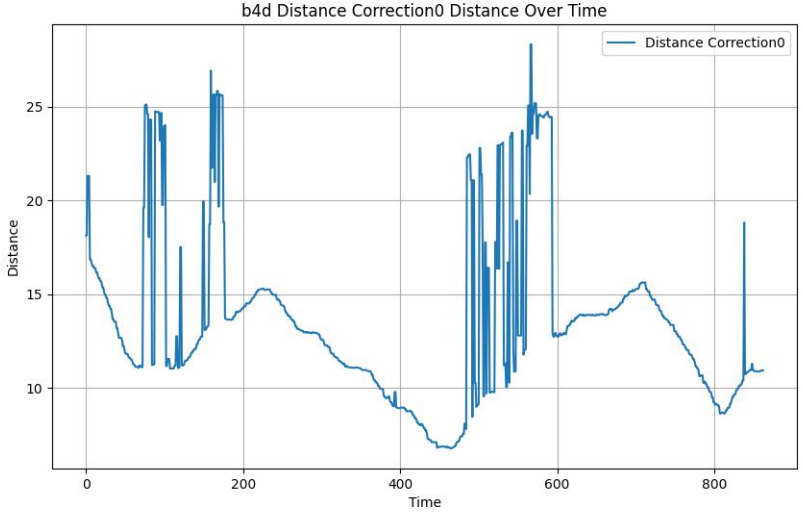


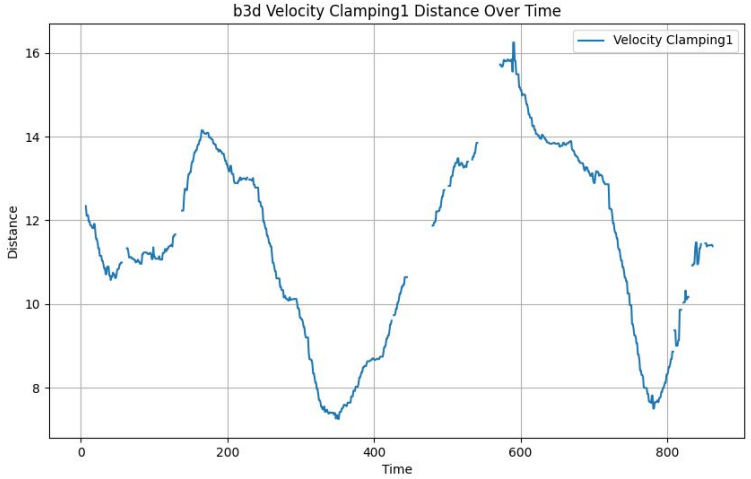




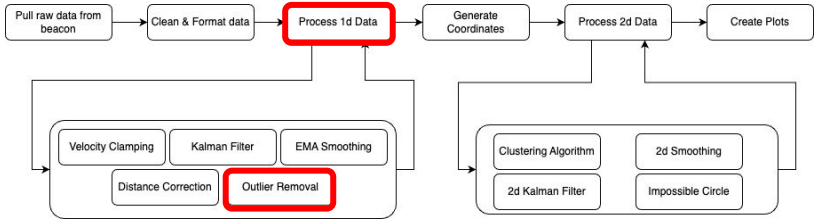
Let’s Process Some Data!

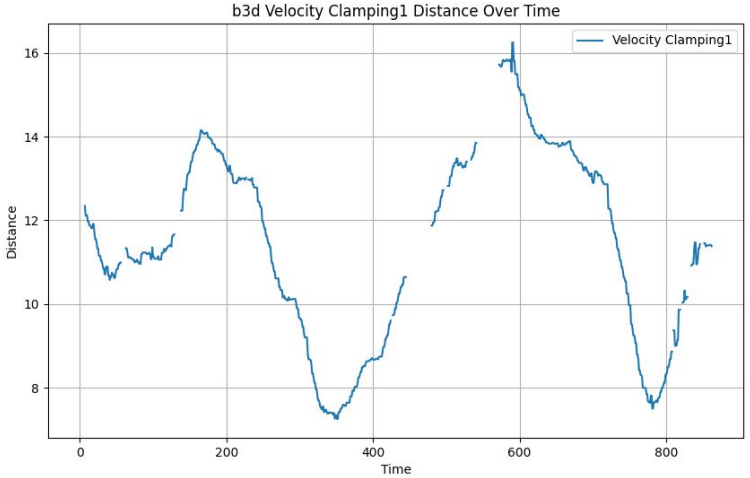


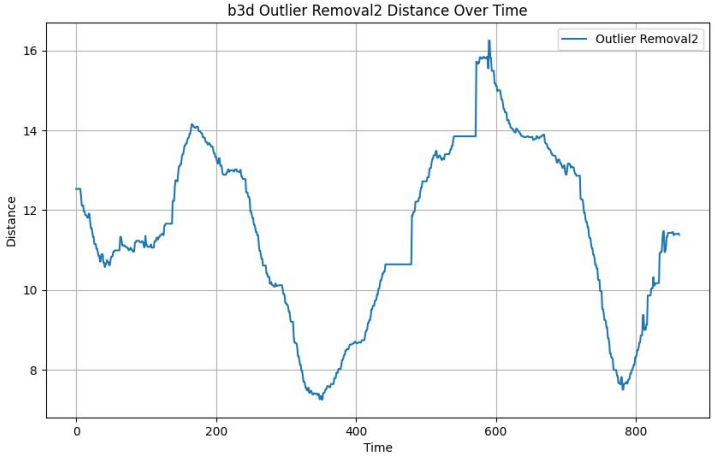




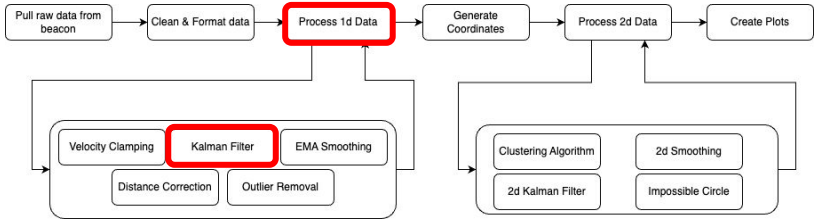
Let’s Process Some Data!

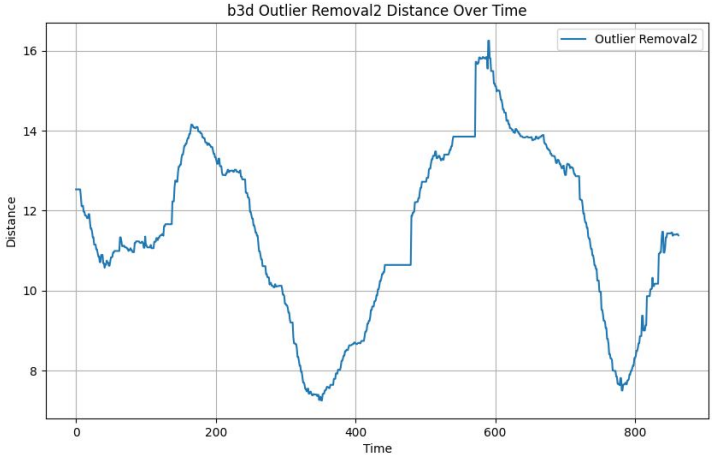


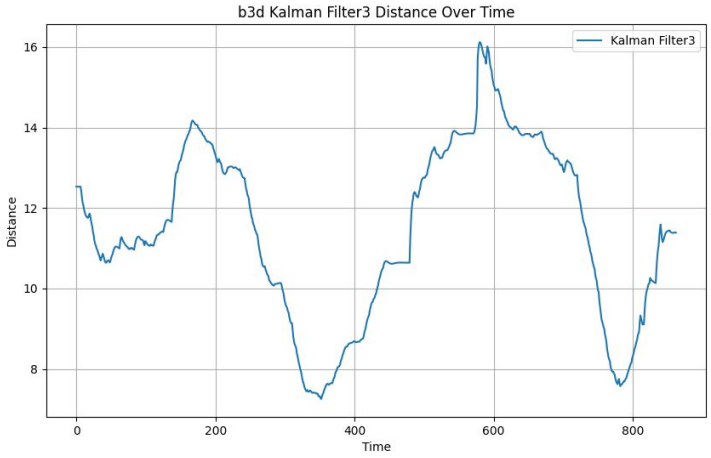




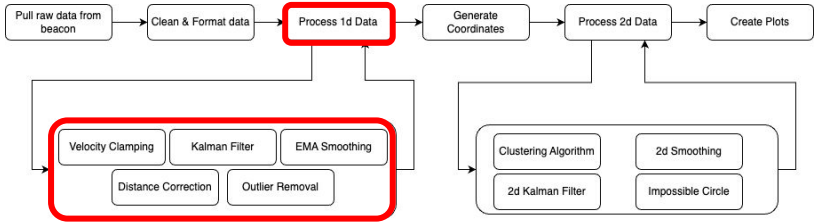
Let’s Process Some Data!



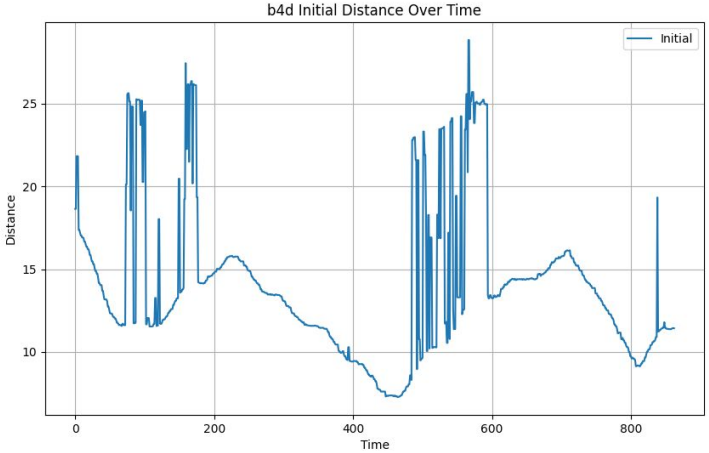




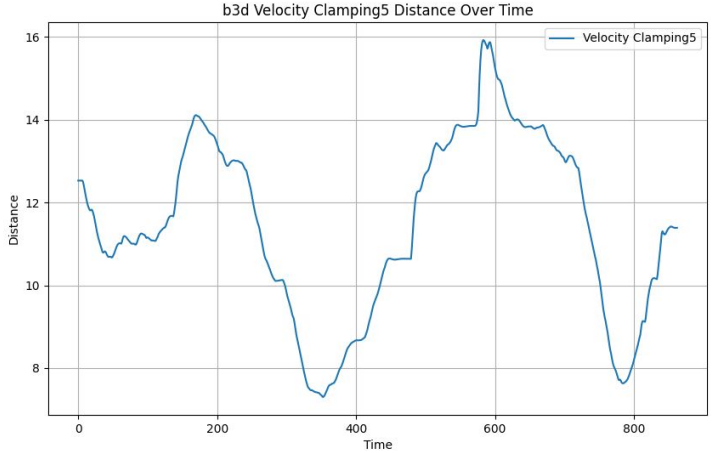
Processing Results



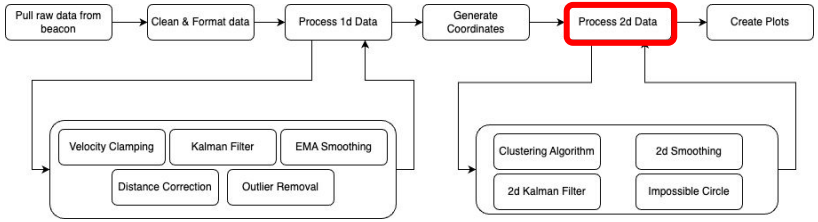
Initial (no processing)

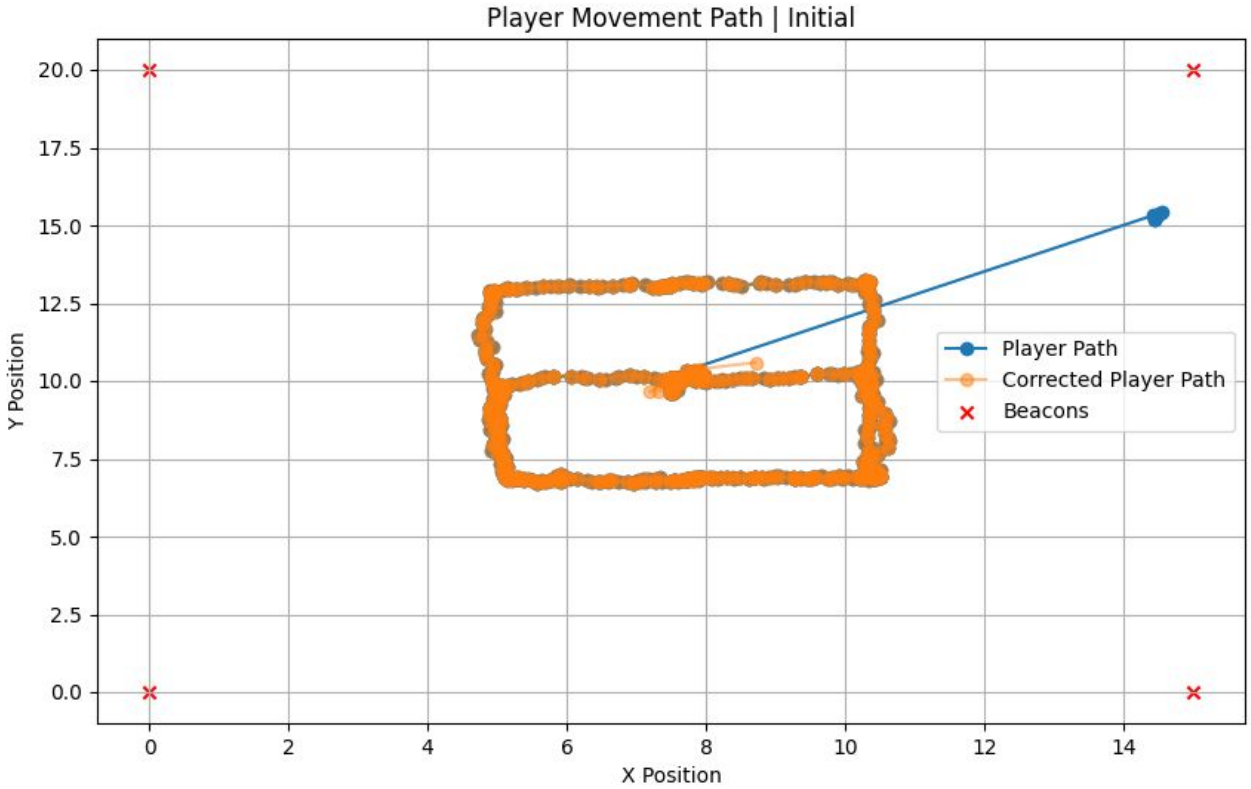


Final (after processing)



Effects of 2d Processing





Final Results/Performance analysis

Show best case (clean data - please do overtheheadtest it is really good), walking with obstacles and more realistic data

Show 1D and 2D plots, comparison to ground truth, MAE

What does our setup do well, what does it do badly?

(mention that if we skip all of the 1D processing we still get a good, sometimes better, result…?)

Final Results

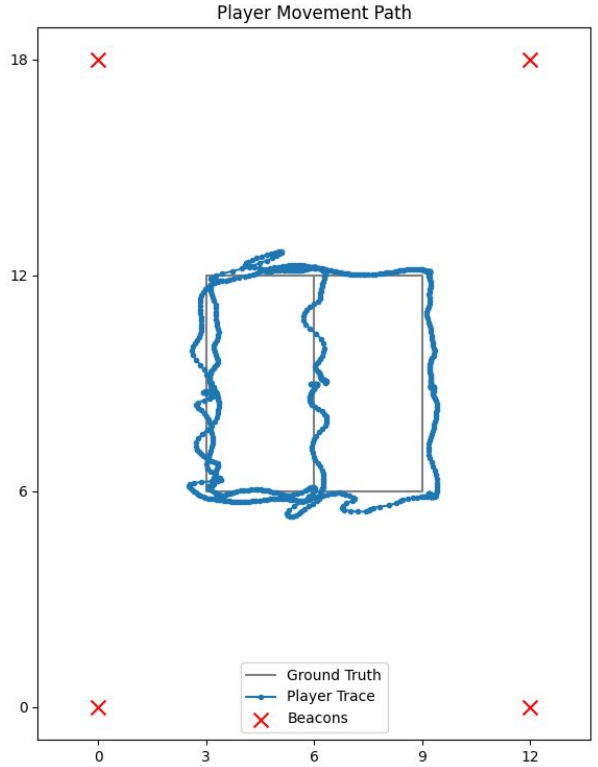
The bad

The good

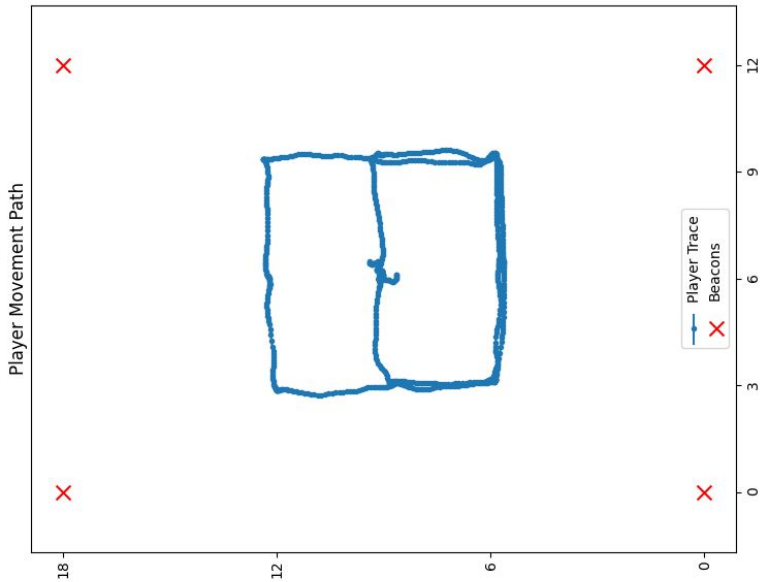
The future

Obstacles are Challenging

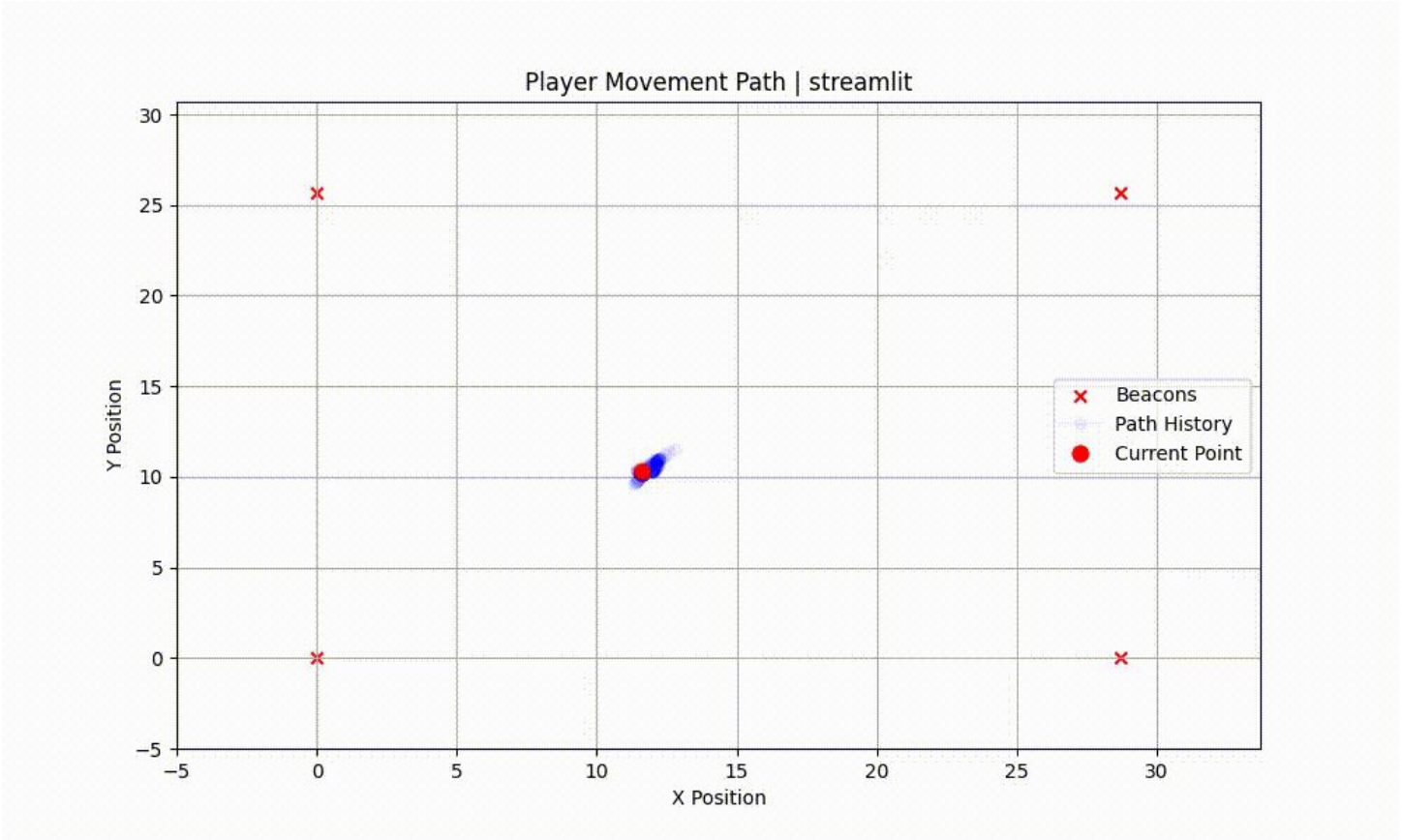
Obstacle Test



No Obstacles

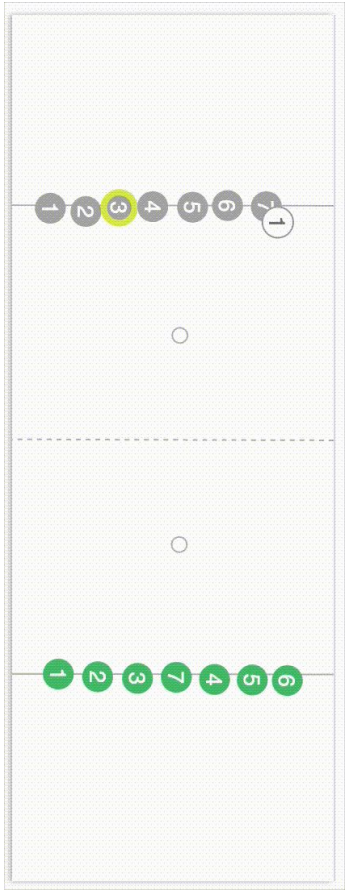


Best Case Plot



Future Work

- Multiplayer
- Frisbee Tracking
- Frisbee Specific Use Cases
- Finish custom UWB modules
- Fine Tuning
- Frontend

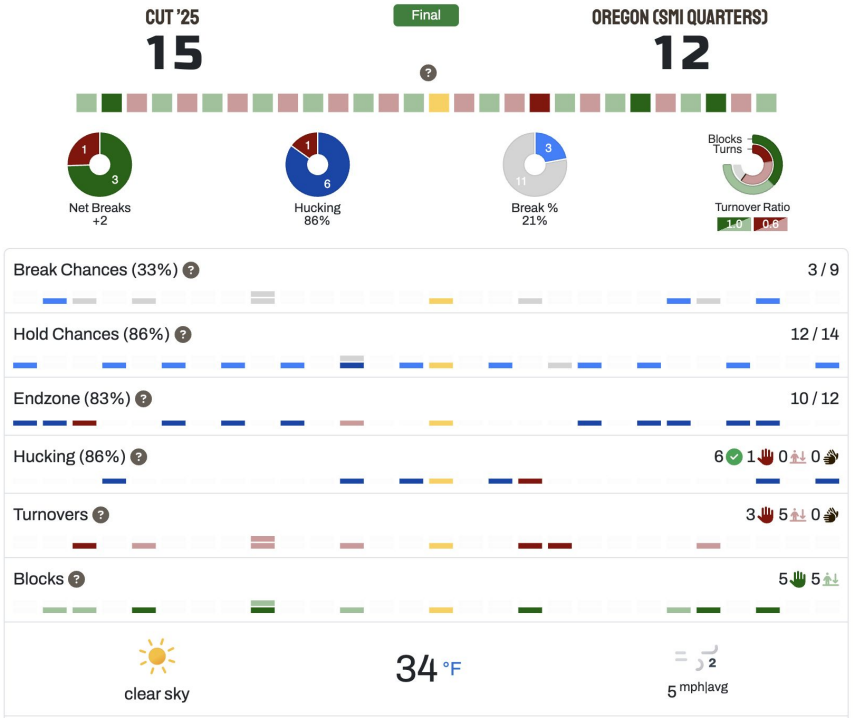


Future Work

- Multiplayer
- Frisbee Tracking
- Frisbee Specific Use Cases
- Finish custom UWB modules
- Fine Tuning
- Frontend

Future Work

- Multiplayer
- Frisbee Tracking
- Frisbee Specific Use Cases
- Finish custom UWB modules
- Fine Tuning
- Frontend



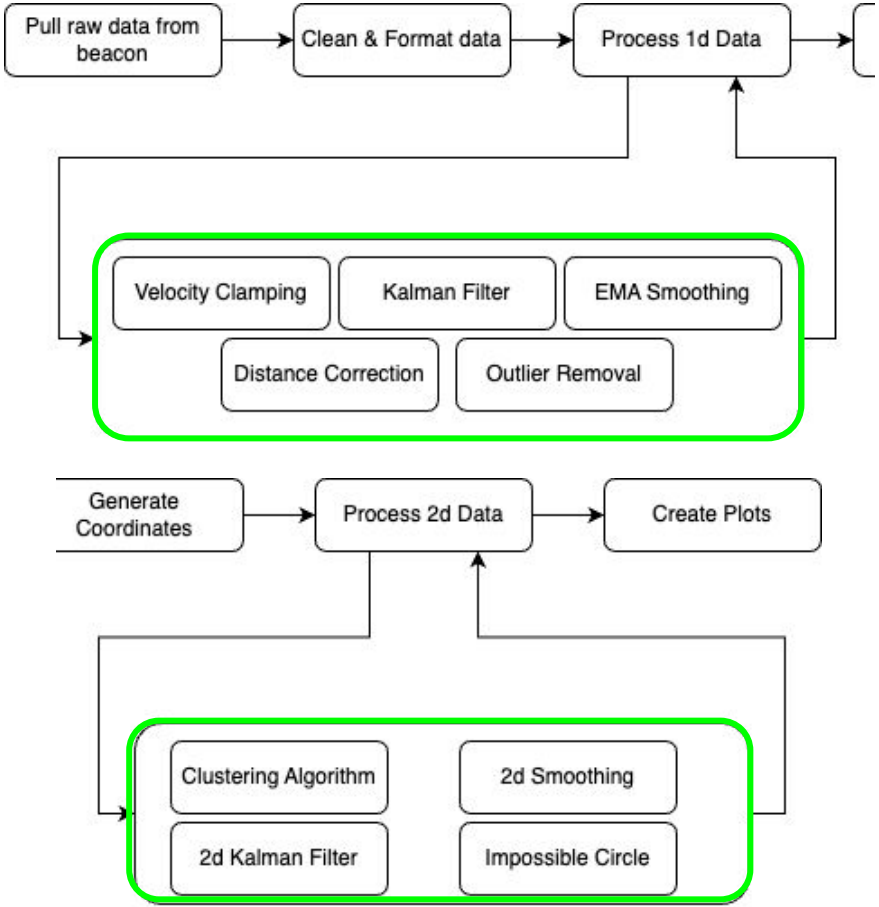
Future Work

- Multiplayer
- Frisbee Tracking
- Frisbee Specific Use Cases
- Finish custom UWB modules
- Fine Tuning
- Frontend

SAVE MONEY!!!

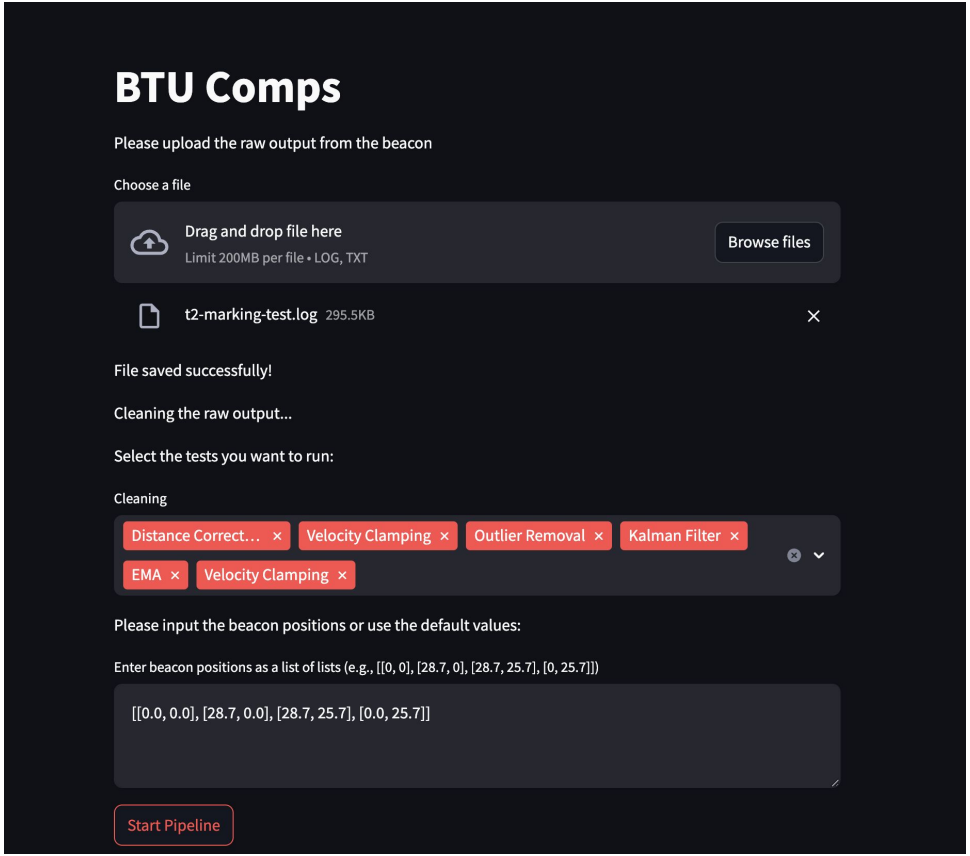
Future Work

- Multiplayer
- Frisbee Tracking
- Frisbee Specific Use Cases
- Finish custom UWB modules
- Fine Tuning
- Frontend



Future Work

- Multiplayer
- Frisbee Tracking
- Frisbee Specific Use Cases
- Finish custom UWB modules
- Fine Tuning
- Frontend



Conclusion

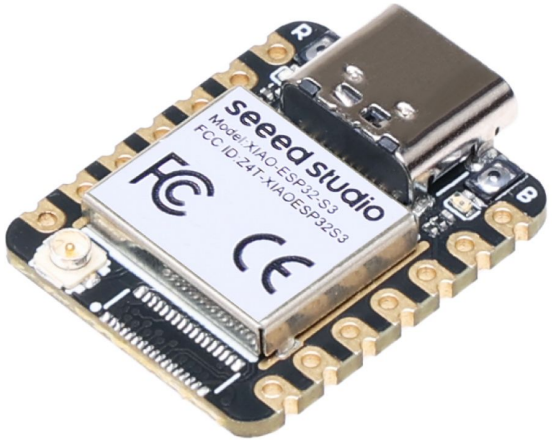
- FTM and UWB are the best technologies we found within a reasonable budget
- FTM is fundamentally not suited to this use-case
  - It can, however, through well-thought-out data processing, it can end up being accurate enough for serious utility, perhaps in a warehouse, for example
  - It also remains significantly cheaper than UWB
- While some data processing is still necessary, UWB is a highly accurate technology that is well-suited to tracking players
  - There are, however, still some difficulties with standard software
  - Prototyping boards are very expensive

Thank you!

Questions?

extras?

Hardware (including wearables)



- Our requirements:
  - **WIRELESS CAPABILITIES**
  - Small
  - Affordable
  - Well supported
    - We don’t have time to write a ton of drivers from

- ESP32 - A modern wireless microcontroller
  - Popular in maker community (shout out makerspace/Aaron)
  - Specifically, we sought boards that supported both wifi and BLE

- Specifically, we decided on the xiao esp32s3 for most* of our testing
  - Tiny
  - Built in charge management
  - Powerful s3 architecture

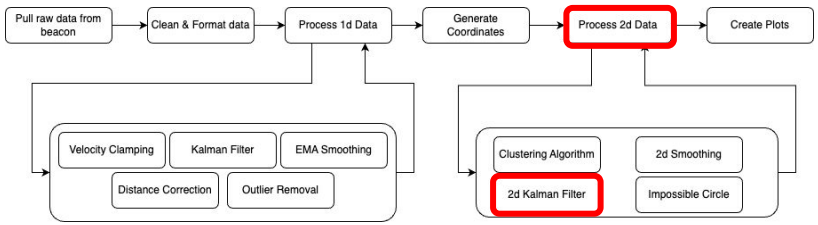
Kalman Filtering

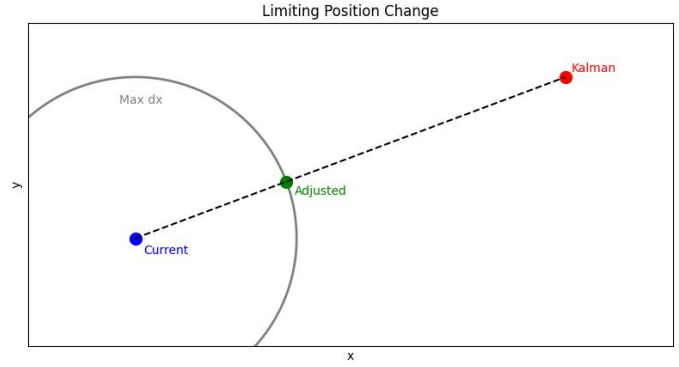
How have we adapted it?

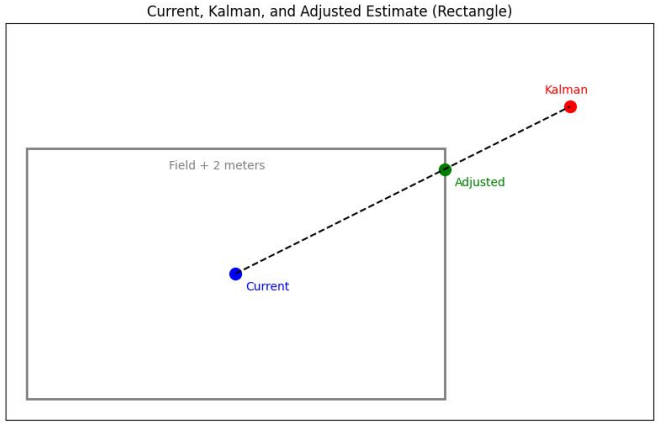
- Kinematic model with constant acceleration

$$x_t = x_{t-1} + vt + \frac{1}{2}at^2$$

- Set an upper limit on change in position based on maximum velocity and measured acceleration
- Use confidence value to dynamically inflate measurement uncertainty (R)
- Limit movement to be within 2 meters of the field

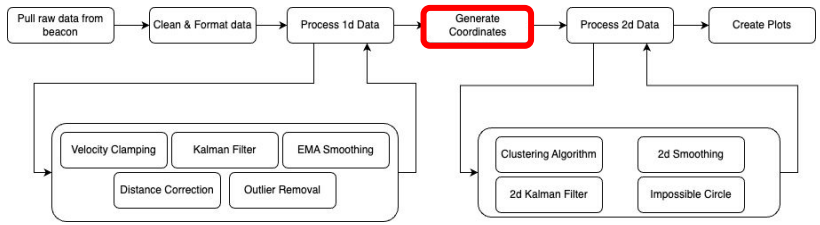


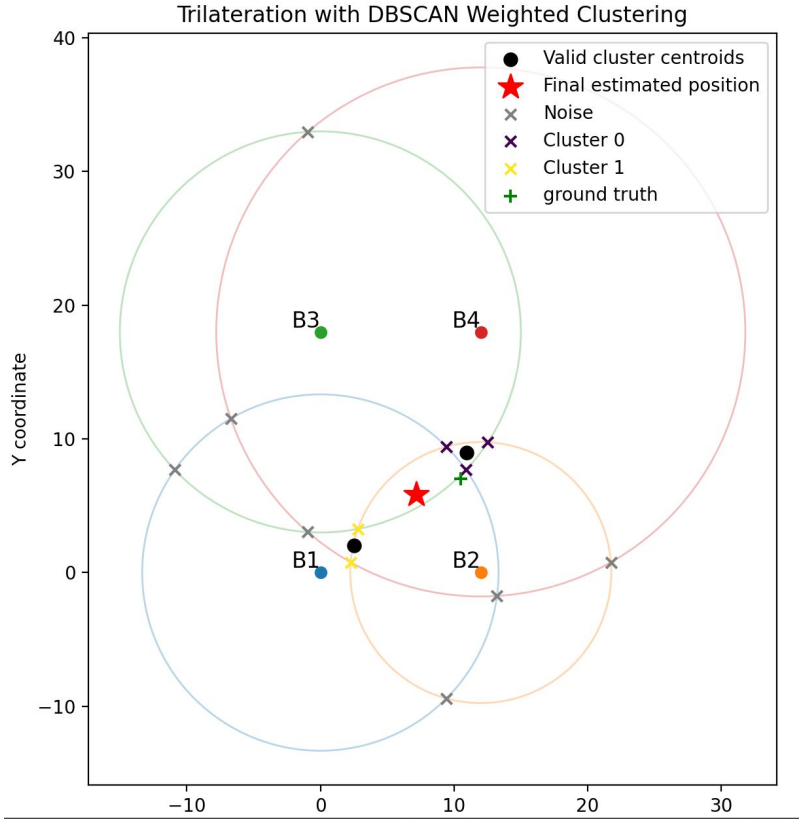




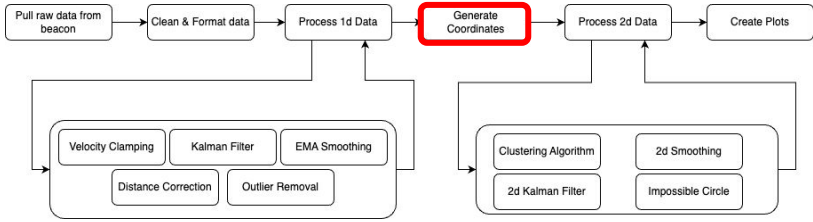
Why only the densest cluster?

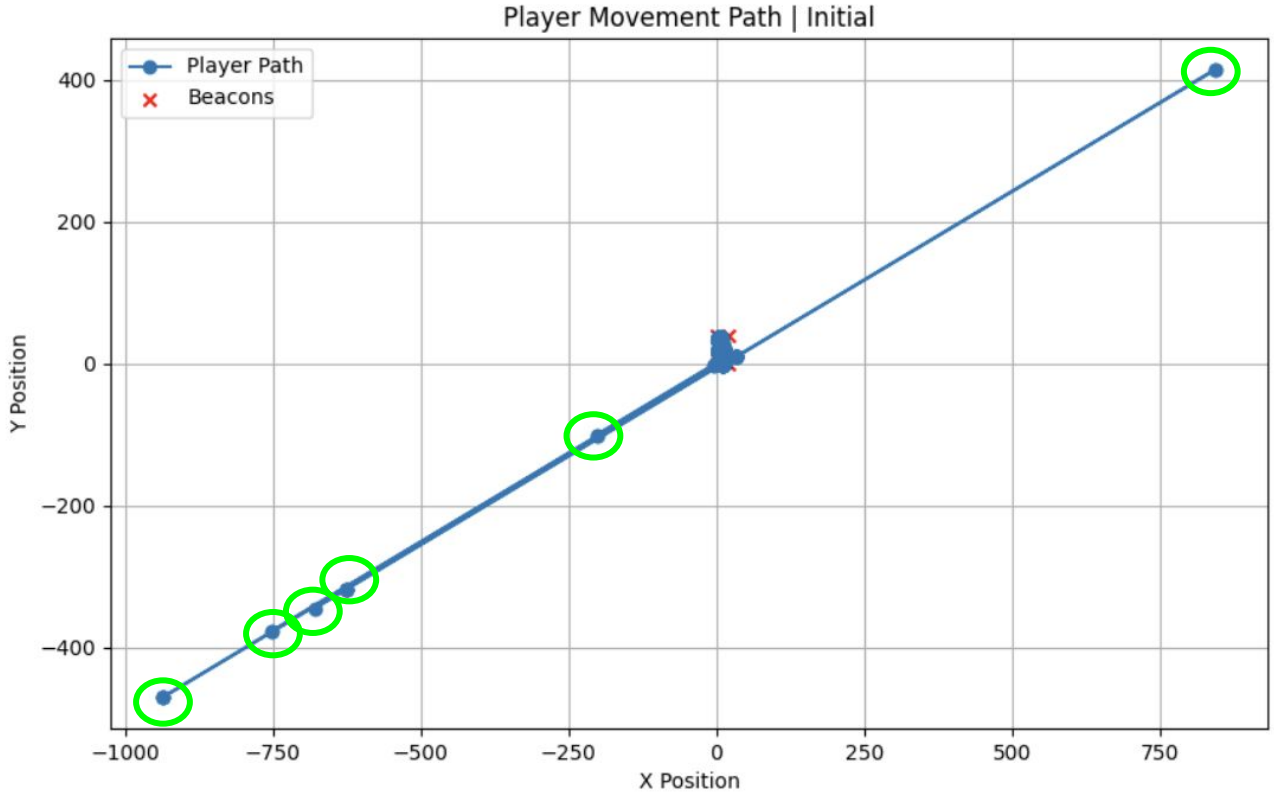
- Could we not use the mean of all intersections to find the coordinate?
- Issue: If one distance is wrong, it can throw off the average
- By using the densest cluster, we choose distance measurements that agree with each other



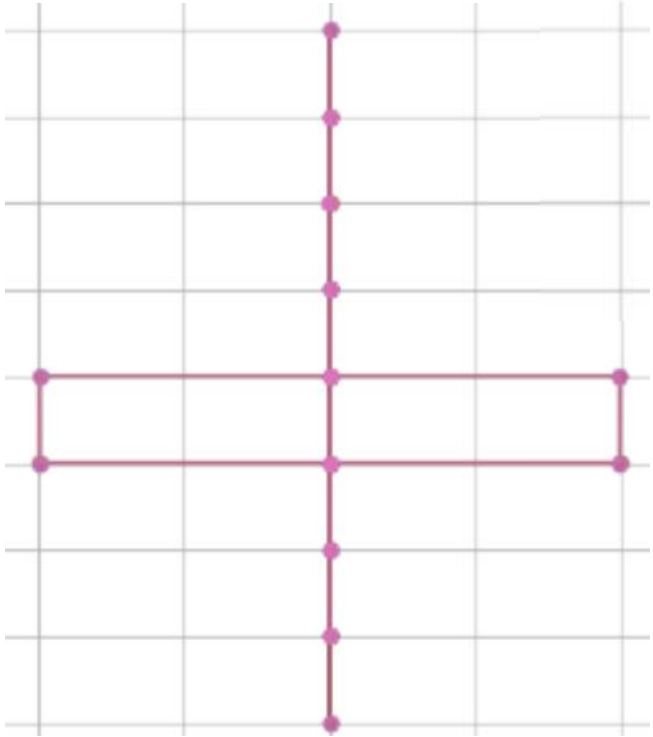


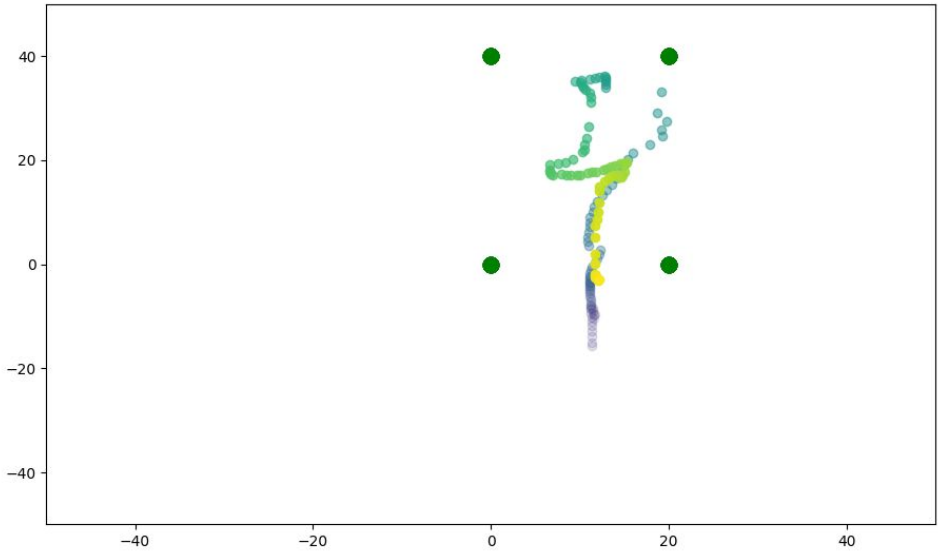
Problems with 1d



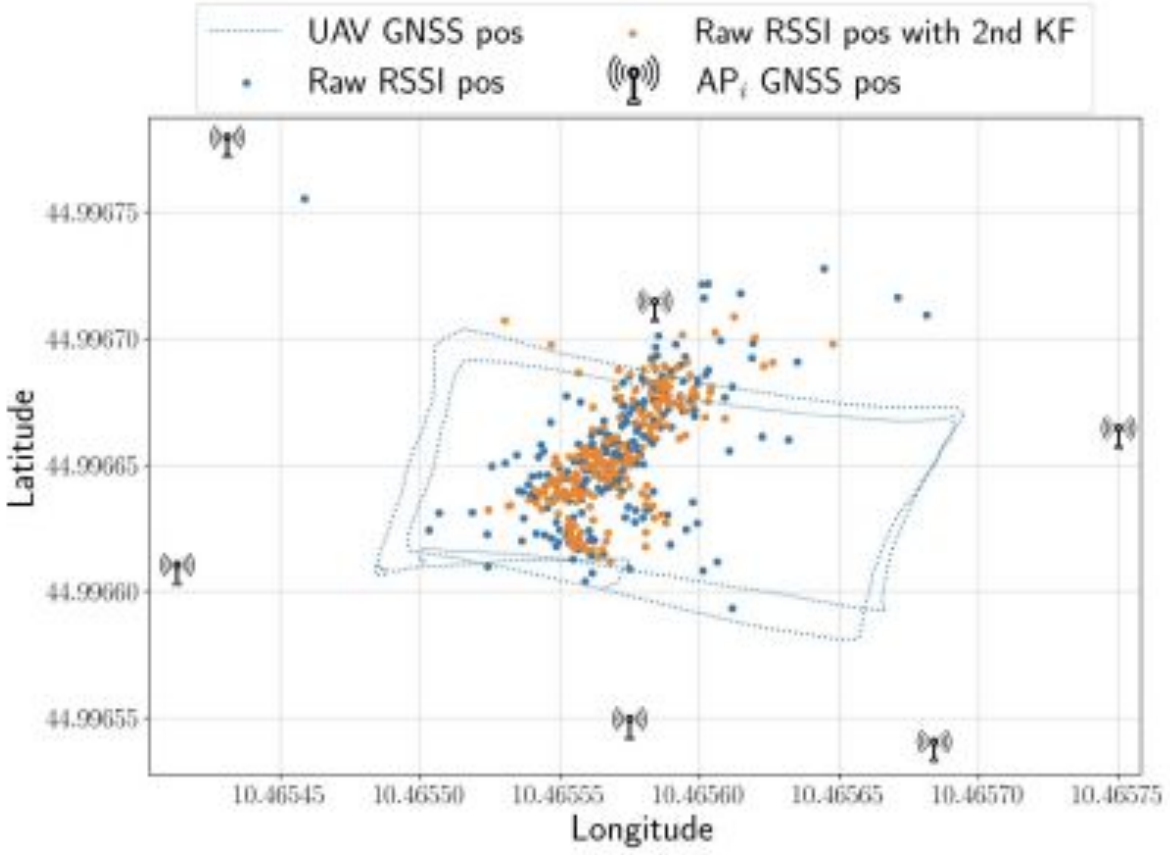


FTM trilateration - best case





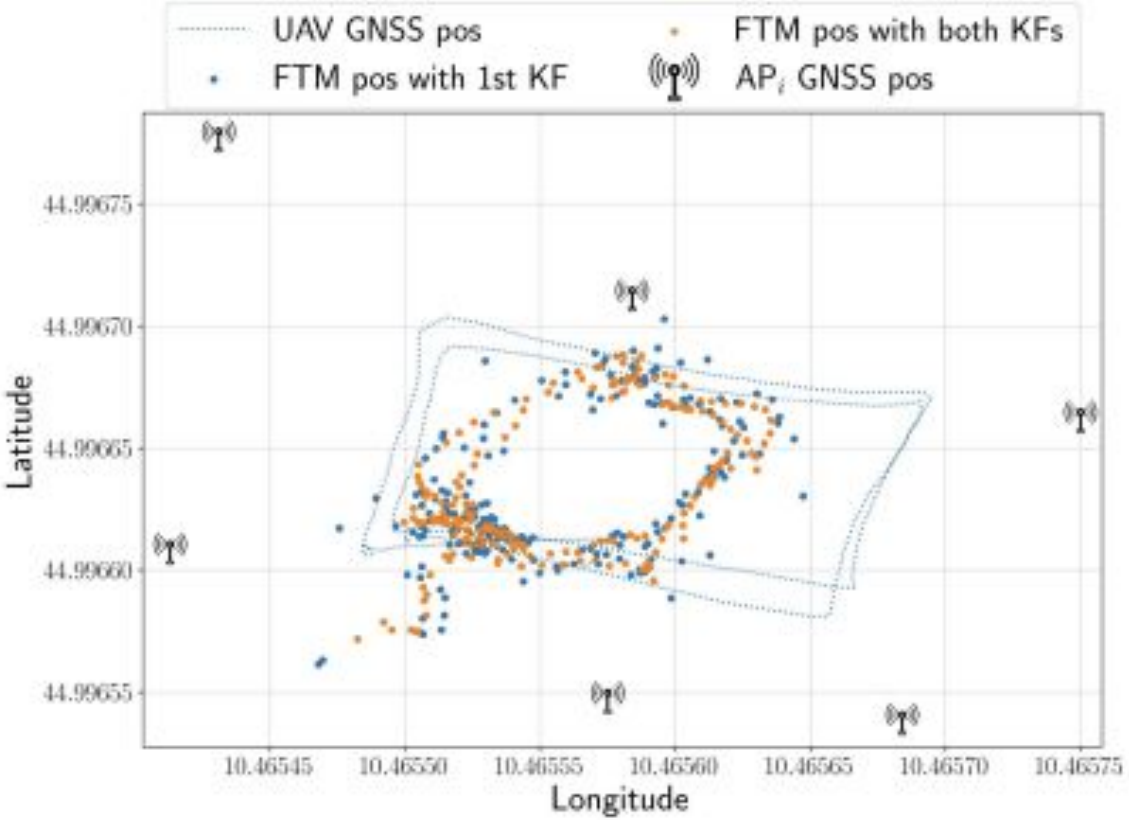
LITERATURE APPROACHES-RSSI



Dotted line = ground truth

Pagliari et al.

LITERATURE APPROACHES-WiFi FTM



Dotted line = ground truth

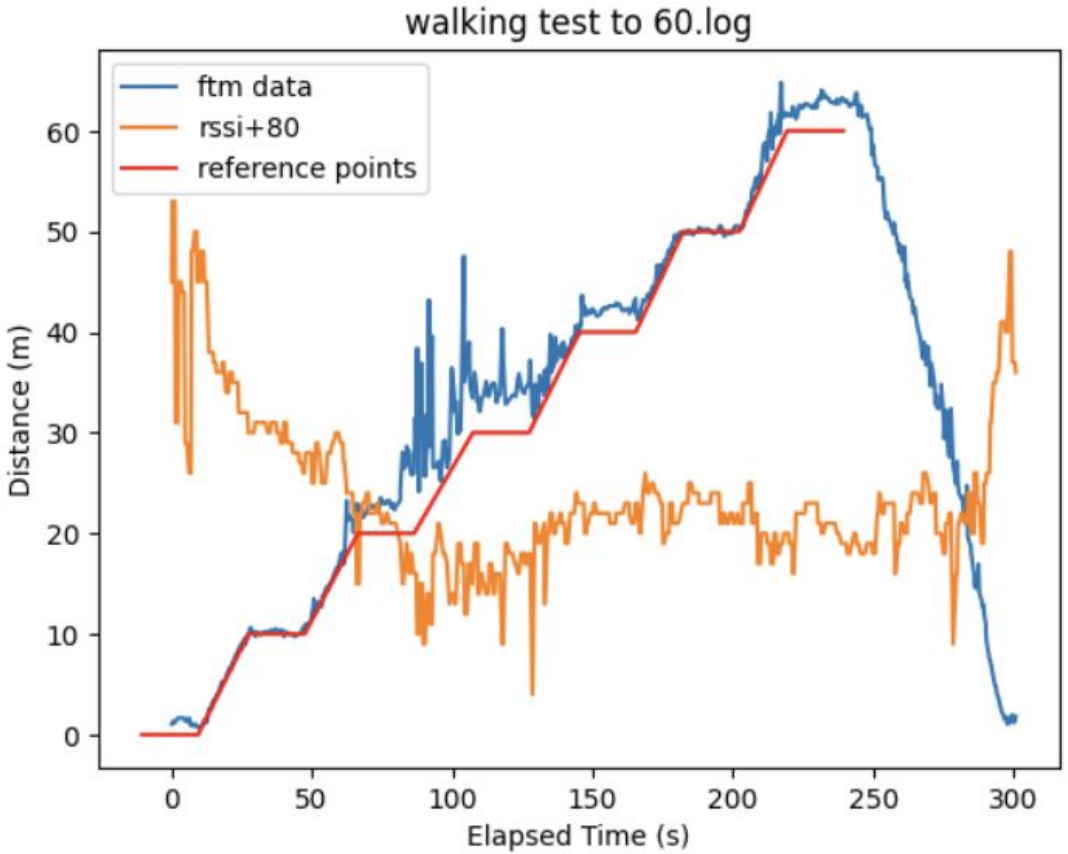
Pagliari et al.

1D ftm - Notice how RSSI plateaus

Baldspot testing:

Walked in straight line away from beacon

Paused at 10m increments



RSSI

RSSI	Signal Strength
> -70 dBm	Excellent
-70 dBm to -85 dBm	Good
-86 dBm to -100 dBm	Fair
< -100 dBm	Poor
-110 dBm	No signal

- Flop
- Extremely noisy signal
- Very challenging to differentiate distances
  - Nonlinear – difficult to convert from RSSI -> distance
  - Low range of measurement