EE 292S
Ege Turan & Patricia Strutz
Fall 2024

Lab 1: Accelerometer, Gyroscope & Tilt, Dead Reckoning

For live gifs:
https://docs.google.com/document/d/1E5fZ6nX09ltbGOVrwGjnuRKZJN6xzdibY4oq_WEFK_4/edit?usp=sharing

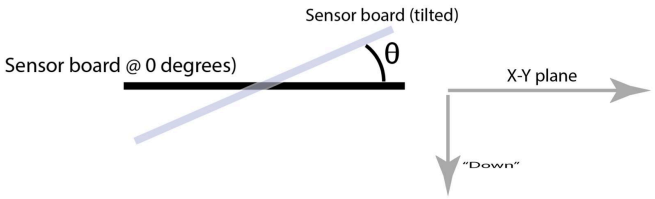
Part 1: Simple Tilt Functionality and Allan Deviation

Part 1.1:

Initial Accelerometer and Gyroscope Measurements

We wrote Python code that implements a simple gravity level with respect to the starting XY plane as described in the spec [Fig. 1]. For raw accelerometer data, we were able to calculate pitch, roll, and yaw by applying trigonometric functions to the accelerometer's x, y, and z-axis values, which are gong to be the effect of gravity on each of the axis, which tells us how it must be oriented. For raw gyroscope data, we did this by integrating the angular velocity over time. We explained the math behind both these methods on the next page. After converting the raw sensor output (for both accelerometer and gyroscope) into readable roll, pitch, and yaw angles, we wanted to test the accuracy of the two separate methods. To do this, we moved our sensor to a ground truth 45˚ angle in each axis. We used a 10 Hz sampling rate for this part. The results are shown in Fig. 2 and Fig. 3.

Fig 1: The definition of tilt



Calculating angles for each of the 3 axes:

a. Using accelerometer

```python
# Function to calculate roll, pitch, and yaw from accelerometer
def calculate_roll_pitch_yaw_accel(accel):
    ax, ay, az = accel
    roll = np.arctan2(-ay, np.sqrt(ax**2 + az**2)) * 180 / np.pi  # Roll angle in degrees
    pitch = np.arctan2(-ax, np.sqrt(ay**2 + az**2)) * 180 / np.pi  # Pitch angle in degrees
    yaw = np.arctan2(ay, ax) * 180 / np.pi  # Approximate Yaw angle in degrees (limited accuracy without magnetometer)
    return roll, pitch, yaw
```

a_x is the accelerometer reading in x-axis.

a_y is the accelerometer reading in y-axis.

a_z is the accelerometer reading in z-axis.

$\text{roll} = \arctan(-a_y / \sqrt{a_x^2 + a_z^2})$

$\text{pitch} = \arctan(-a_x / \sqrt{a_y^2 + a_z^2})$

$\text{yaw} = \arctan(a_y / a_x)$

For yaw, this is a very crude and unreliable estimate.

b. Using gyroscope

```python
# Function to integrate gyro data to estimate roll, pitch, and yaw
def integrate_gyro(gyro, dt, current_angles):
    roll, pitch, yaw = current_angles
    # Convert angular velocity from degrees/second to radians/second
    gyro_rad = np.radians(gyro)

    roll += gyro_rad[0] * dt * 180 / np.pi  # Integrate roll

    pitch += gyro_rad[1] * dt * 180 / np.pi  # Integrate pitch

    yaw += gyro_rad[2] * dt * 180 / np.pi    # Integrate yaw
```

```
    # Normalize angles to be within -180 to 180
    roll %= 360
    pitch %= 360
    yaw %= 360

    roll = roll if roll < 180 else roll - 360
    pitch = pitch if pitch < 180 else pitch - 360
    yaw = yaw if yaw < 180 else yaw - 360

    return roll, pitch, yaw
```

Gyroscope readings: ω_x, ω_y, ω_z (angular velocities around the x, y, and z axes, respectively, in degrees per second).

Time interval: $\Delta t = dt$

Current angles: (ϕ,θ,ψ) (\phi, \theta, \psi) roll, pitch, and yaw, respectively.

Running integrate to update roll, pitch, and yaw:

$\phi_{n+1} = \phi_n + \omega_{x,n} * \Delta t$

$\theta_{n+1} = \theta_n + \omega_{y,n} * \Delta t$

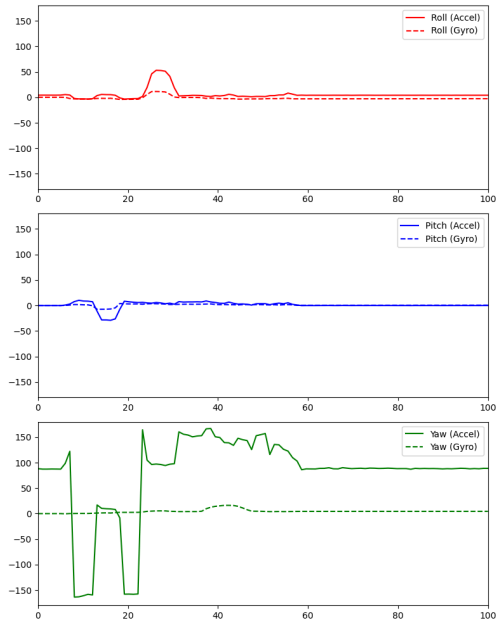
$\psi_{n+1} = \psi_n + \omega_{z,n} * \Delta t$

Deriving tilt:

To do a net tilt calculation as in [Fig. 1], we can compute the angles in the x and y planes (pitch and roll). We calculate:

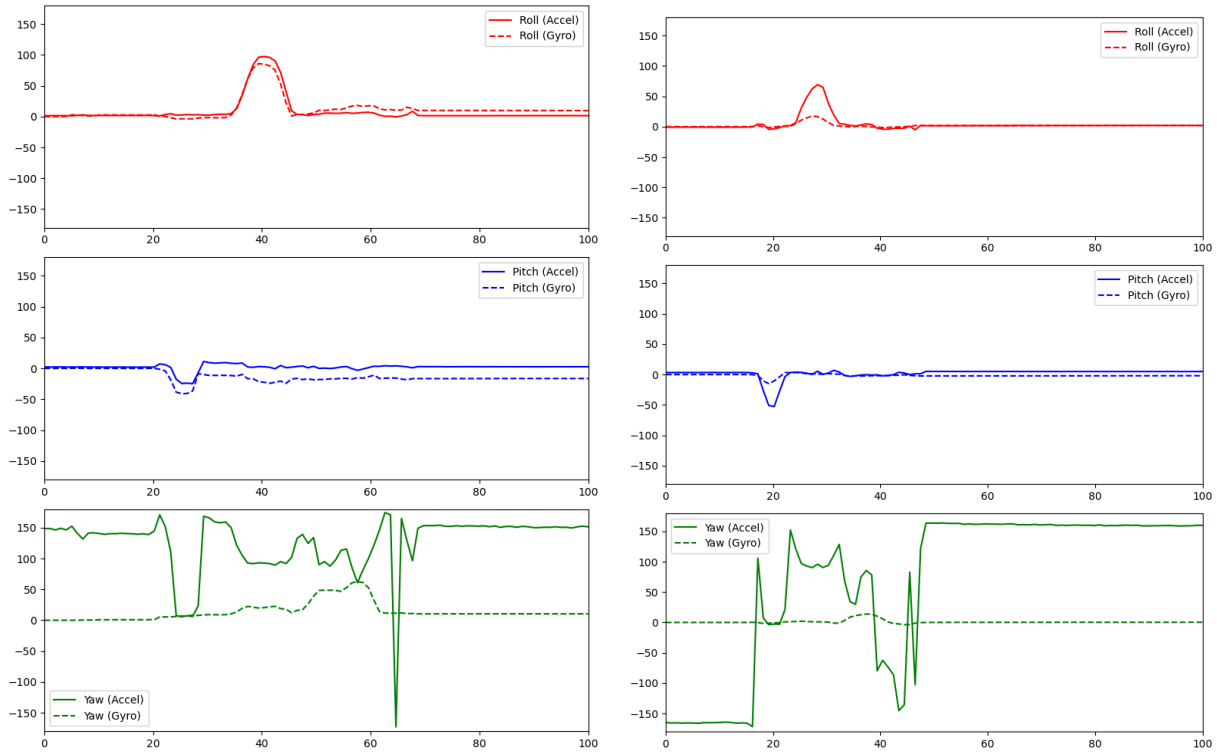
$$tilt = \sqrt{pitch^2 + roll^2}$$

Fig 2: Experiments to determine the accuracy of accelerometer versus the gyroscope in determining the angles in our 3 axes, degrees versus time in seconds



After seeing similar results over multiple repetitions of this experiment, we concluded that in pitch and roll, the accelerometer produced more accurate results while the gyroscope tended to underestimate the true angle. In yaw, however, the accelerometer proved completely useless. This likely has to do with the gyroscope being erroneous after multiple timesteps due to integration errors, while the accelerometer struggles to measure yaw due to the gravity vector not changing significantly in this instance. For this reason, accelerometer data will have less long term error, while the gyroscope measurements are only useful short term (before integration errors add up).

Fig. 3: Two more experiments, going to a ground truth angle of 90˚, degrees versus time in seconds

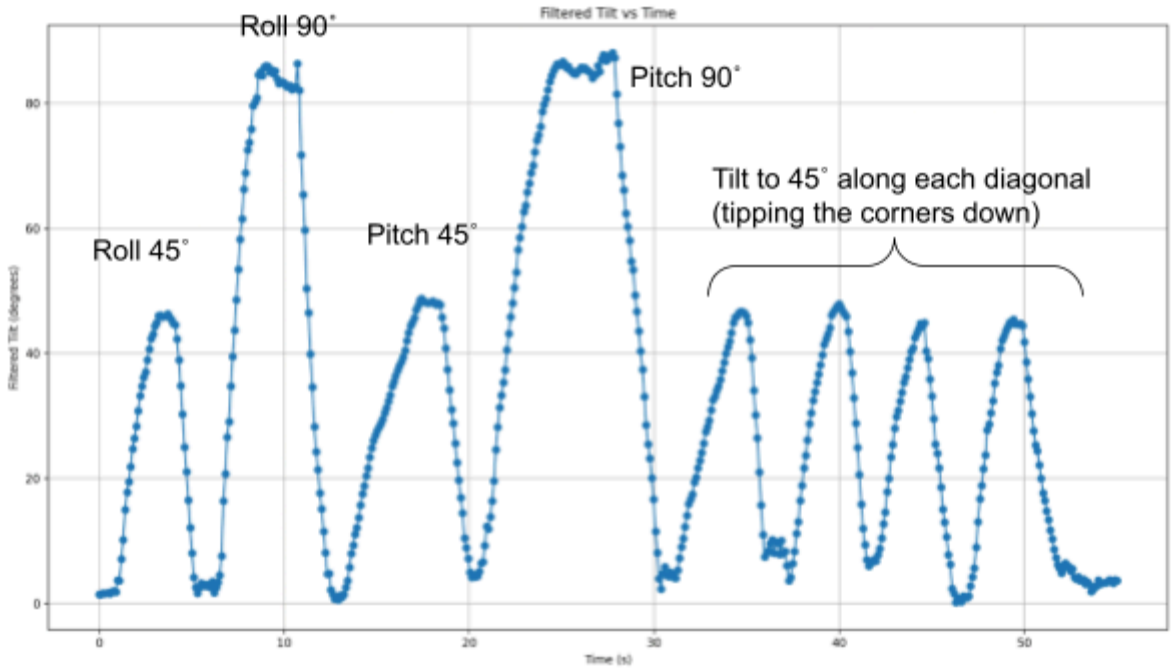


Fusion Strategy and Results

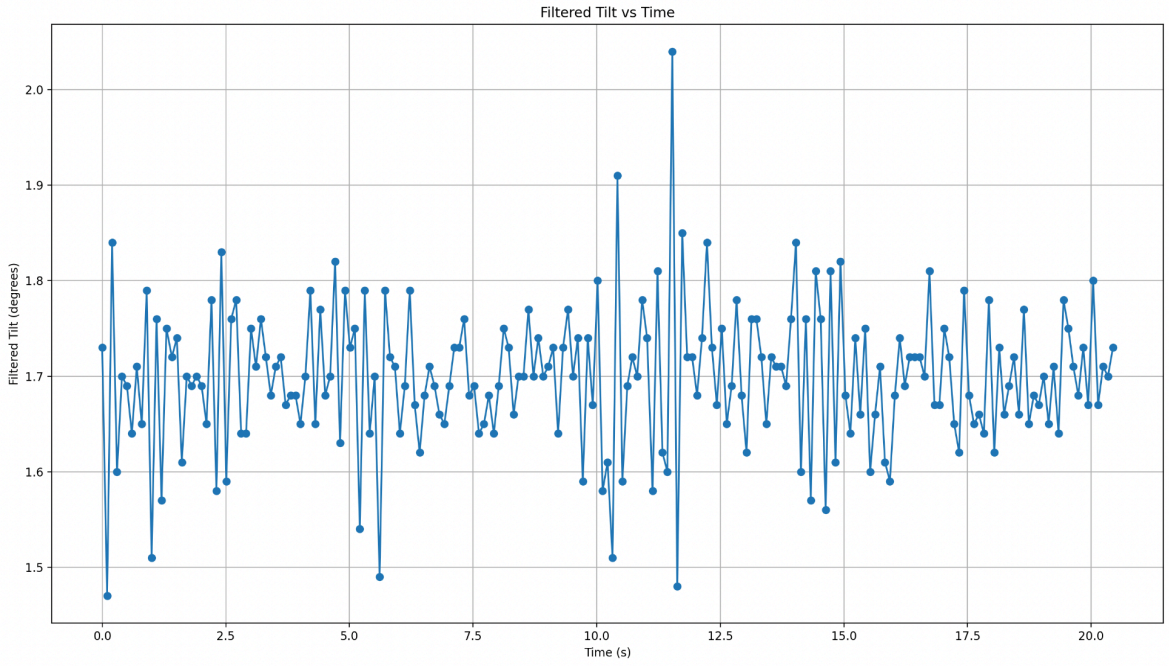
After this observation, we chose to fuse our sensor measurements through a weighted average of accelerometer and gyroscope measurements, favoring accelerometer (with weight 0.98, leaving a 0.2 contribution for the gyroscope) for pitch and roll, and favoring gyroscope (with weight 1, completely ignoring the accelerometer) for yaw. The results shown in Fig. 4 demonstrate the accuracy of our fused sensor.

Fig. 4: a) Tilting to 45˚ and 90˚ angles in different directions. b) Sensor completely stationary. c) Full rotations, followed by figure 8 movement.

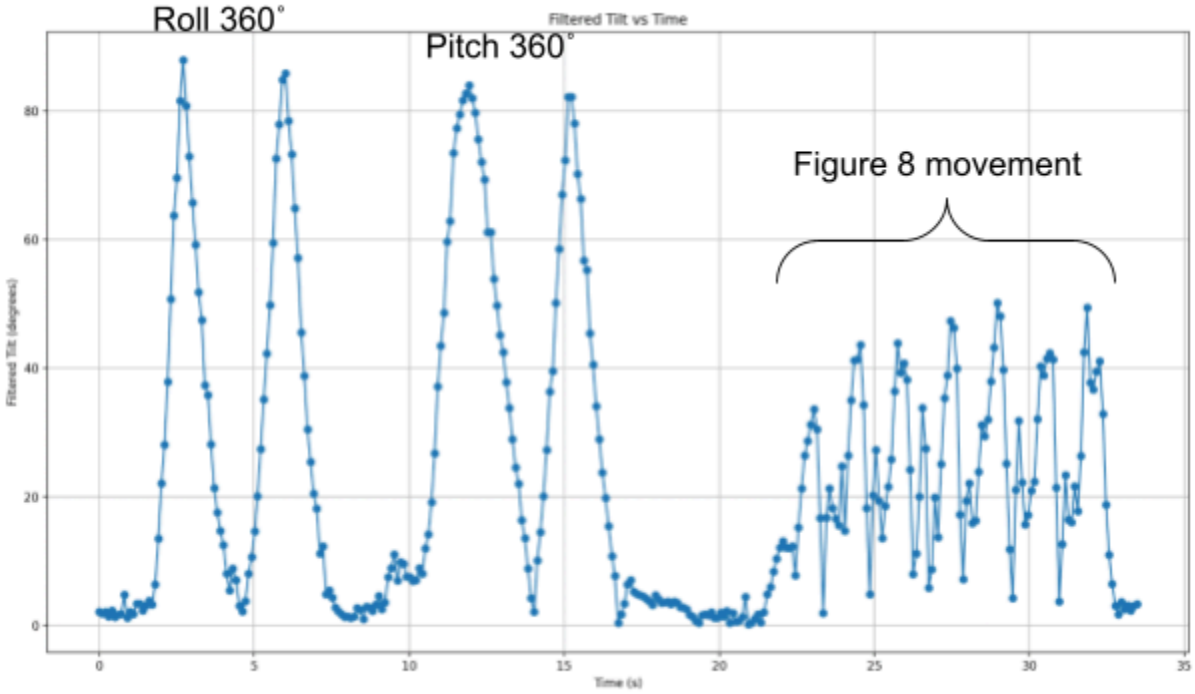
a)


Filtered Tilt vs Time
Roll 90˚
Pitch 90˚
Tilt to 45˚ along each diagonal
(tipping the corners down)
Roll 45˚
Pitch 45˚
Filtered Tilt (degrees)
Time (s)

b)


Filtered Tilt vs Time
Filtered Tilt (degrees)
Time (s)

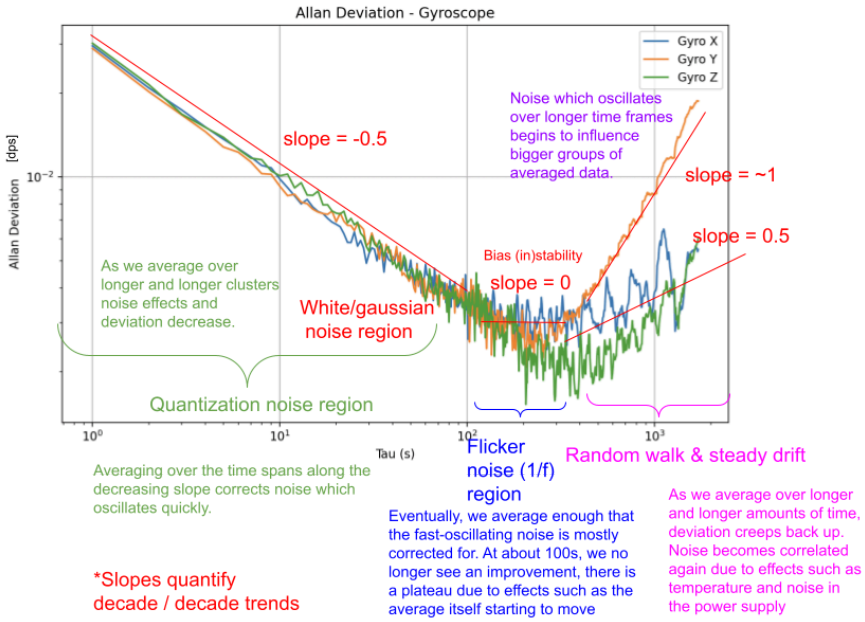
c)

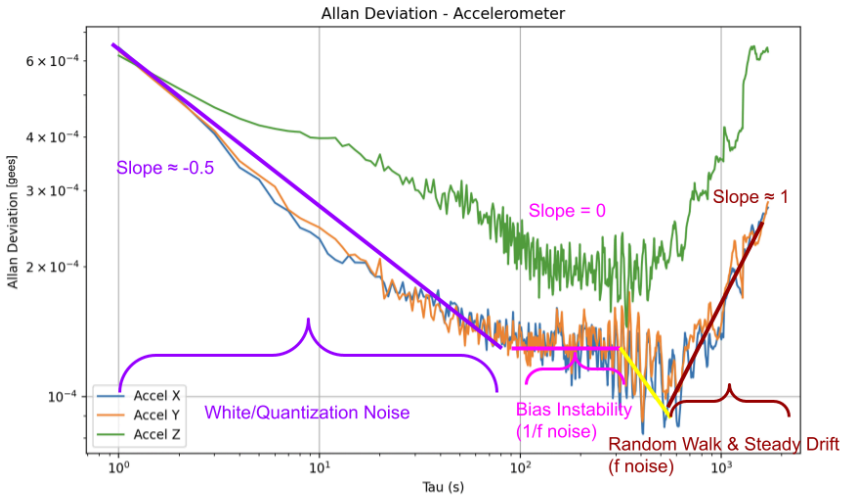


Part 1.2: Allan deviation plots for raw accelerometer and gyroscope data

For the Allan deviation plot, we collected 30 minutes of data from the completely stationary sensor in a room at ~26˚C [Fig. 5].

Fig. 5: Allan deviation plots for a) gyroscope and b) accelerometer.





For the accelerometer, we saw relatively similar results for measurements in the x and y directions, while the z-direction data had more noise. For the sake of the graph, only the x and y data points were specifically labeled but the regions hold for the z-direction as well. The first region of the graph shows a decreasing white noise with increasing tau - this makes sense, as this random, independent noise is averaged out with larger values of tau. The bias instability region reflects the near minimal noise level, at which point averaging does not show significant

improvement anymore and flicker noise is most present. In the final region, we see increased noise again as random walk processes such as bias drift and slow, frequency dependent fluctuations (f noise) create cumulative effects that increase for larger values of tau.

Part 2: Implementing a 1-D Kalman filter to determine position based on acceleration measurements in a single axis (X-axis only)

1-D Kalman Filter

For Part 2, we increased sampling rate to 100Hz after getting unstable results at 10Hz. To build our Kalman Filter, we tried 2 different approaches:

a) Assuming constant acceleration; state vector includes position, velocity, and acceleration; sensor data is inputted as measurement

b) Assuming constant velocity; state vector includes position and velocity; acceleration is inputted as a control input with a control input matrix to describe effect on position and velocity

The limitation of (b) was that we could not account for noise variance in our measurements; ultimately, we chose option (a). Thus, our 1-D Kalman Filter consisted of the following matrices:

```
kf.F = np.array([[1, dt, 0.5*dt**2],
                 [0, 1, dt],
                 [0, 0, 1]])          kf.H = np.array([[0, 0, 1]])

kf.x = np.zeros(3)  kf.P = np.eye(3)*1e-2   kf.Q = np.eye(3)

meas_var = 1e2
kf.R = np.array([[meas_var]])
```

Bias Calibration

In order to reduce the bias in our x-axis sensor measurements that is introduced through gravity in the case of a slight roll or pitch, we added a 5 second calibration period to our data collection script. Since gravity is large, it overpowered noise during our calibration step. Moreover, during this time, the acceleration measurements along each axis were recorded, then averaged over the time period, to also remove any noise skewing in a certain direction. The bias was subtracted from the following sensor measurements. For simplicity, we saved raw sensor data to a file before processing it using the Kalman filter. For collecting our data, we used a ruler to measure out 6ft to use as a ground truth measurement.

Kalman Filter Results and Adjustments to Reduce Errors

In the Kalman filter, the acceleration measurements were fed in one at a time as measurement inputs. We adjusted the measurement variance, process noise covariance, and initial covariance matrix to achieve the most accurate results. As seen in Fig. 6, this produced somewhat reasonable results, but the position estimate was severely impacted by a small positive bias from the velocity prediction not returning fully to 0 when stationary, creating a huge error over time. One possible explanations for this phenomenon is an asymmetry in the Kalman acceleration prediction, such that integration of acceleration does not go to 0, causing a bias velocity to remain. This could be, for instance, due to the smoothing of the peak acceleration.

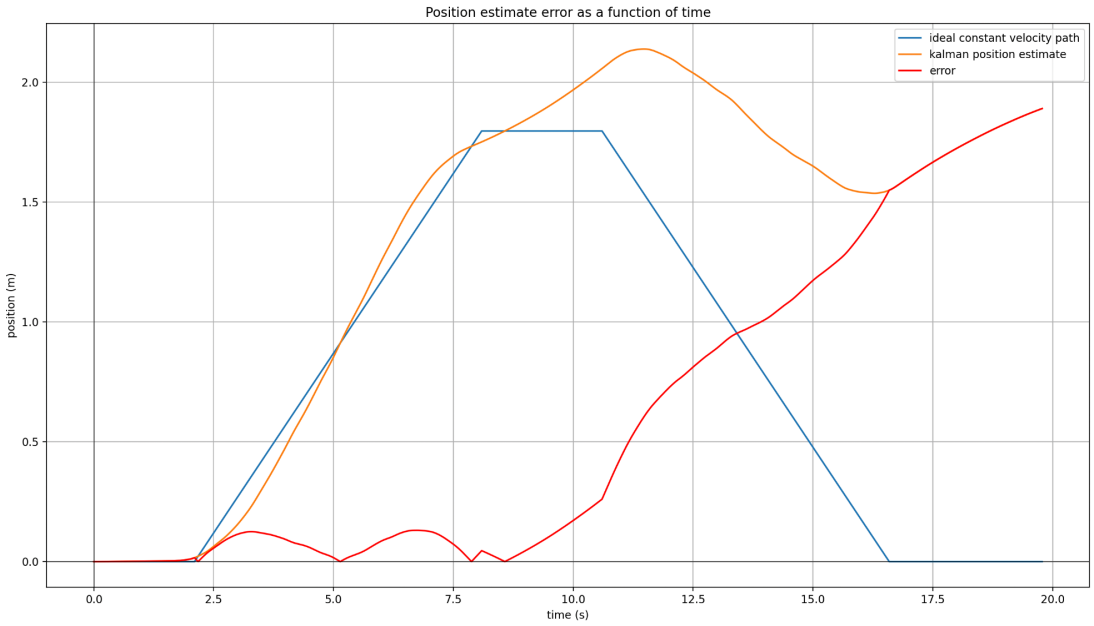
To resolve this issue, we introduced a small clamp into our measurements: if the measured acceleration was below 0.1 m/s^2, we reported to the Kalman Filter that the measurement was 0. We got this 0.1 m/s^2 threshold by observing the drift over many experiments; from Fig 7., we can see that the stagnation of acceleration below 0.1 m/s^2 at about 0.07 m/s^2 from 8.1s through 10.6s, where it is supposed to be 0 causes a large skew. This dramatically reduced the rate at which error accumulates, see Fig. 7. We tried a third approach to see if we could eliminate the bias entirely: still clamping acceleration, and additionally damping velocities when below a threshold value[1]. This approach sacrifices accuracy for more precision and stability - as the estimated peak distance traveled is less than the ground truth value [Fig. 8].

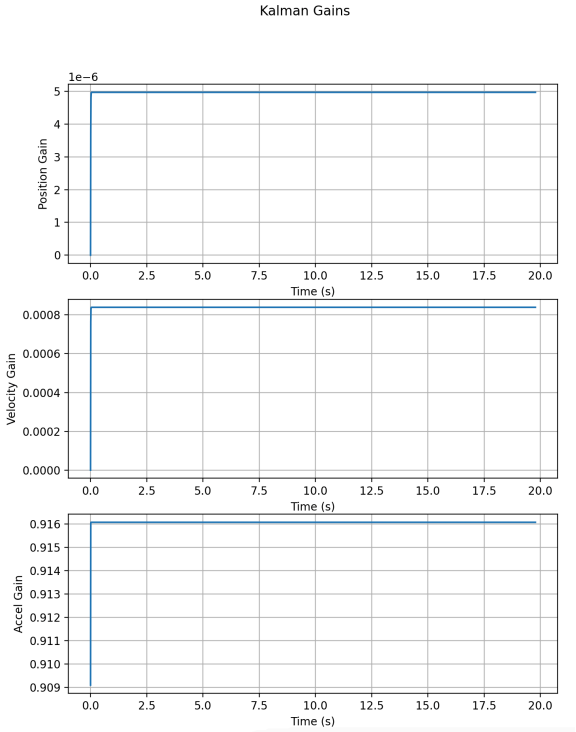
Error Calculation

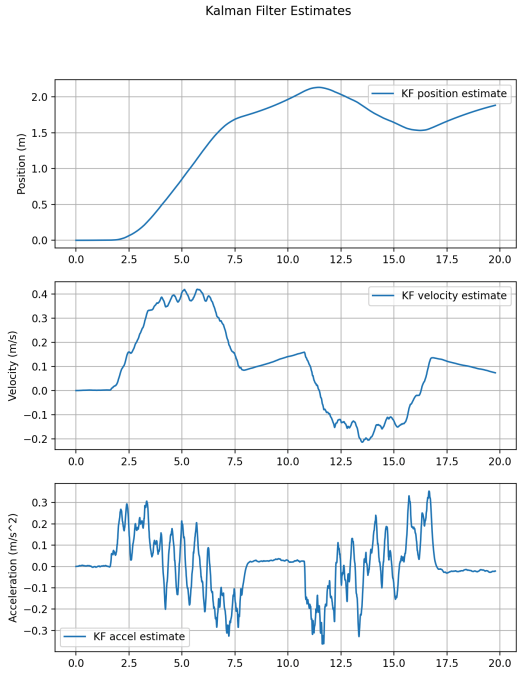
[1] in practice, this meant for |velocity| < 0.05, velocity = (velocity * 0.9) at each iteration

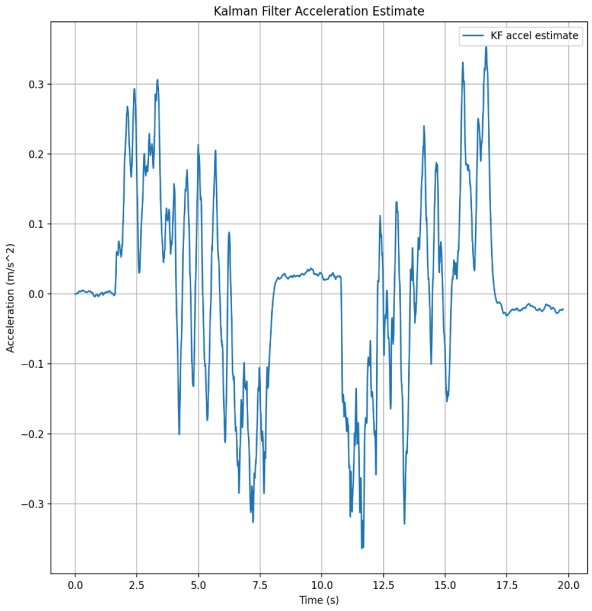
In all cases, we calculated error by assuming a constant velocity between the stationary beginning position and the stationary peak distance, then back at the same rate. While this is not directly a ground truth measurement, it is the ideal movement we wished to capture. From there, we took the absolute value of the difference in Kalman prediction and ground truth, and plotted this value as our error.

Fig 6: 6ft, fast speed, there-and-back in a nearly-straight line (no minimum-clamping on accel)




Kalman Gains
1e−6
Position Gain
Time (s)
Velocity Gain
Time (s)
Accel Gain
Time (s)


Kalman Filter Estimates
KF position estimate
Position (m)
KF velocity estimate
Velocity (m/s)
KF accel estimate
Acceleration (m/s^2)


Kalman Filter Acceleration Estimate
KF accel estimate
Acceleration (m/s^2)
Time (s)

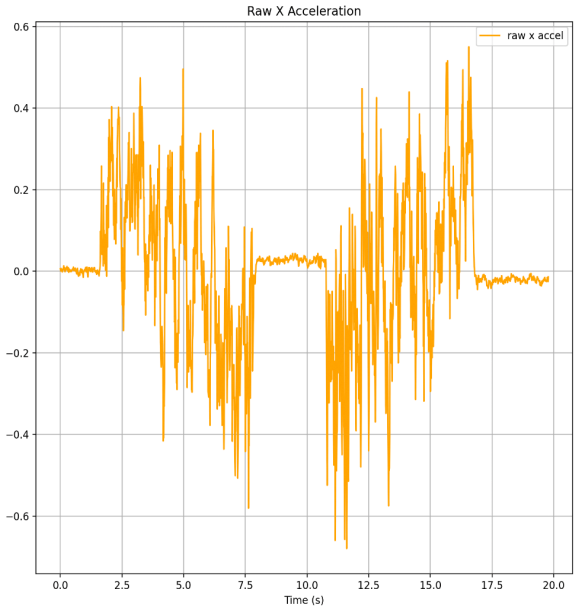
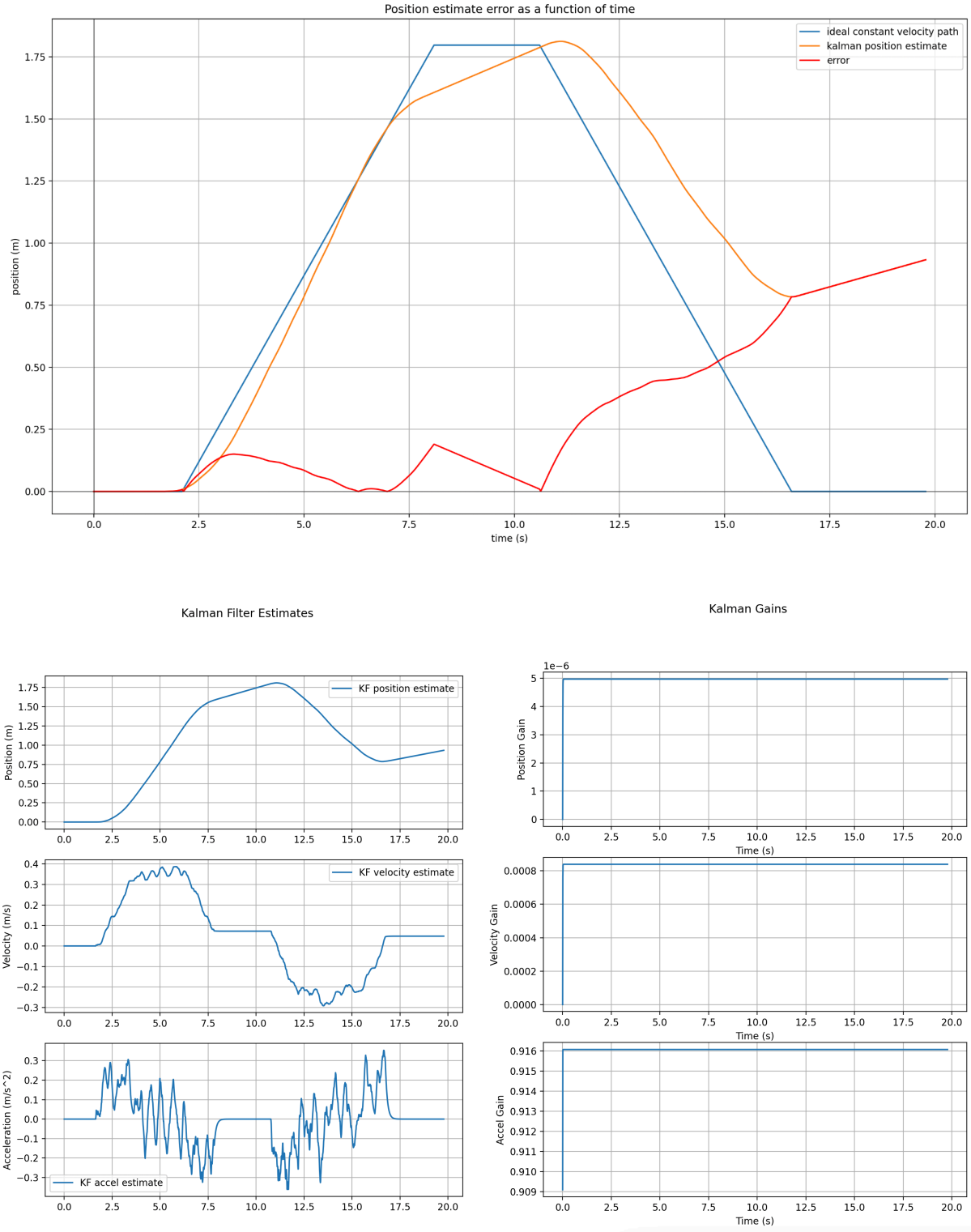
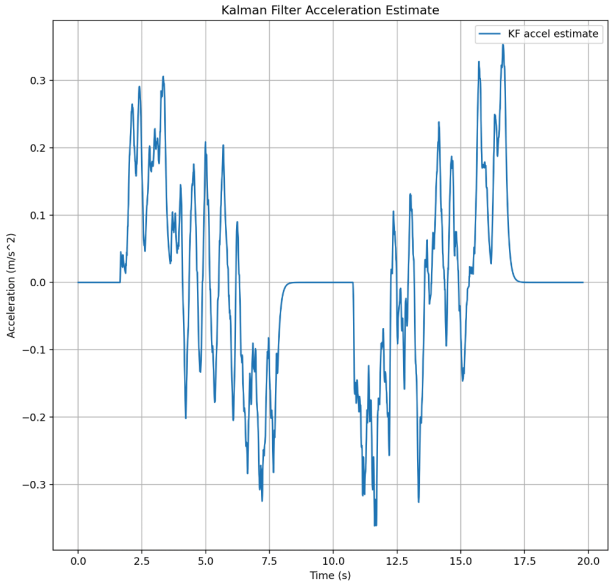

Raw X Acceleration
raw x accel
Time (s)

Fig 7: 6ft, fast speed, there-and-back in a nearly-straight line (minimum-clamping on accel)




Kalman Filter Acceleration Estimate
KF accel estimate
Acceleration (m/s^2)
Time (s)

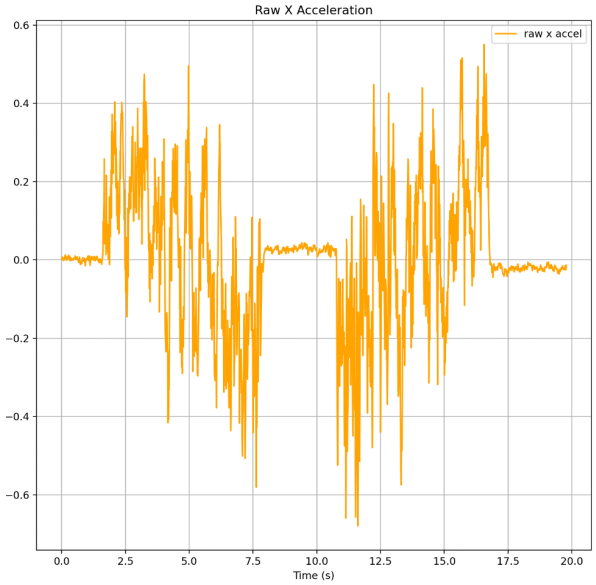
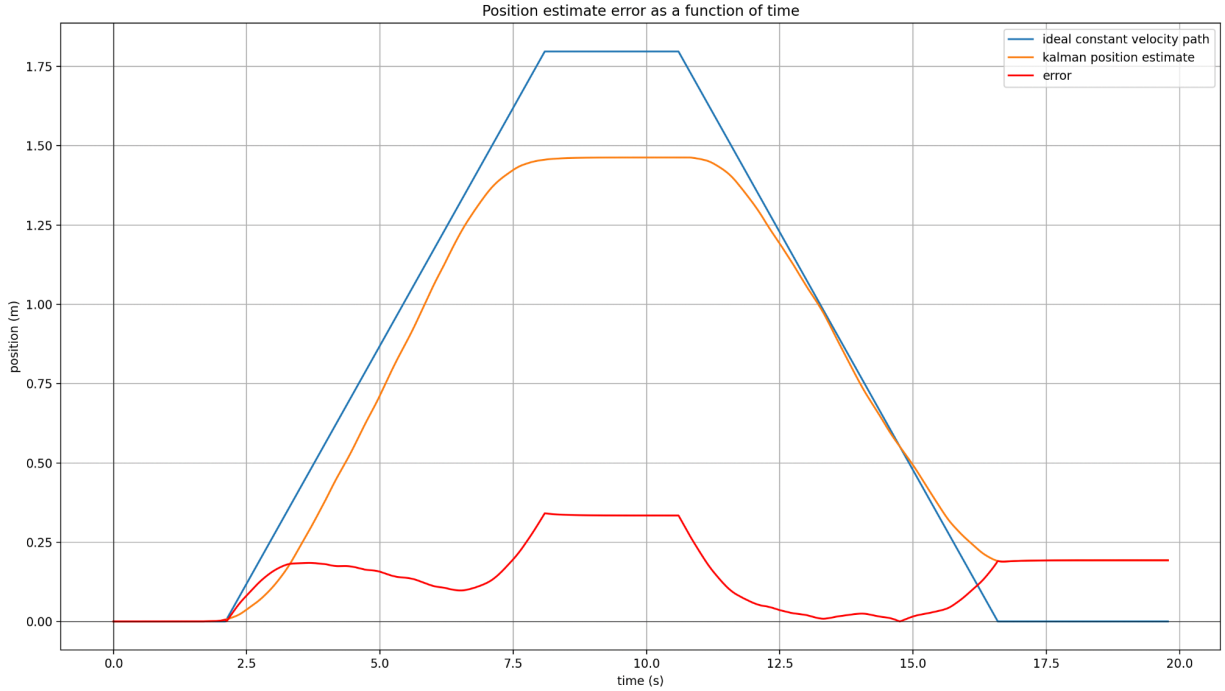
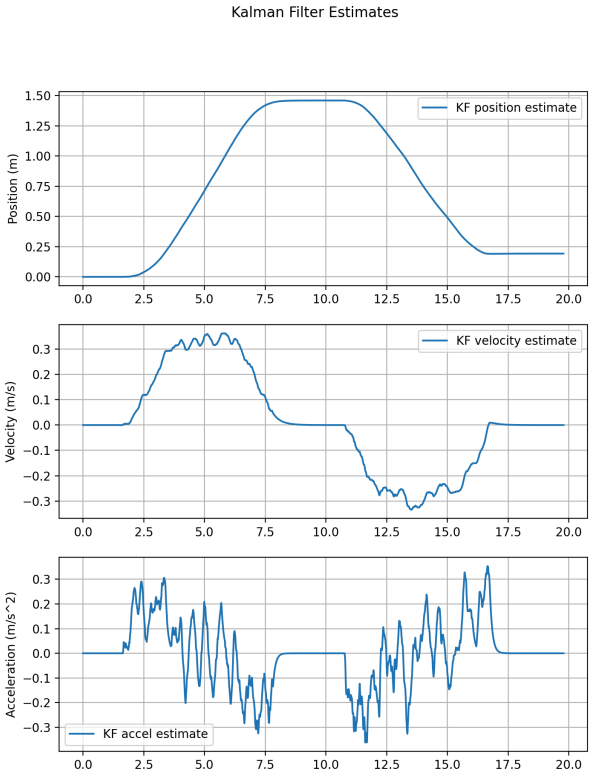
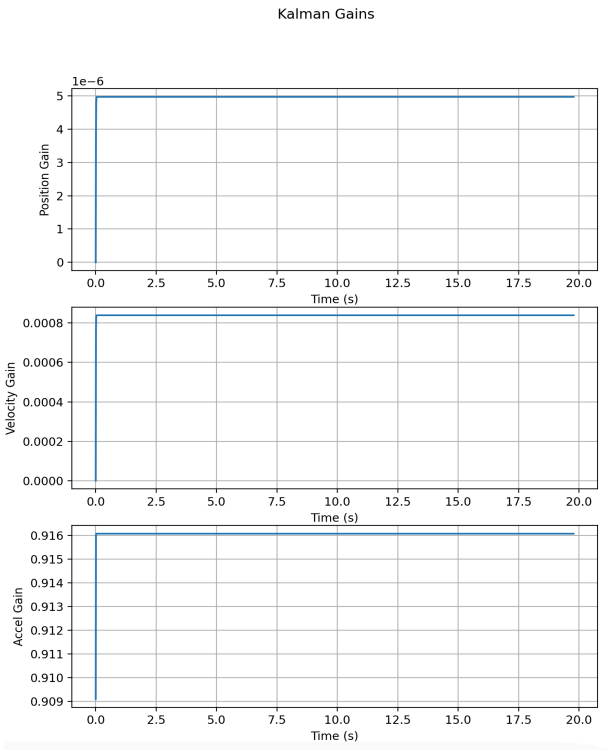

Raw X Acceleration
raw x accel
Time (s)

Fig 8: 6ft, fast speed, there-and-back in a nearly-straight line (minimum-clamping on accel and diminishing velocity bias for low velocities; this approach sacrifices accuracy for more stability)





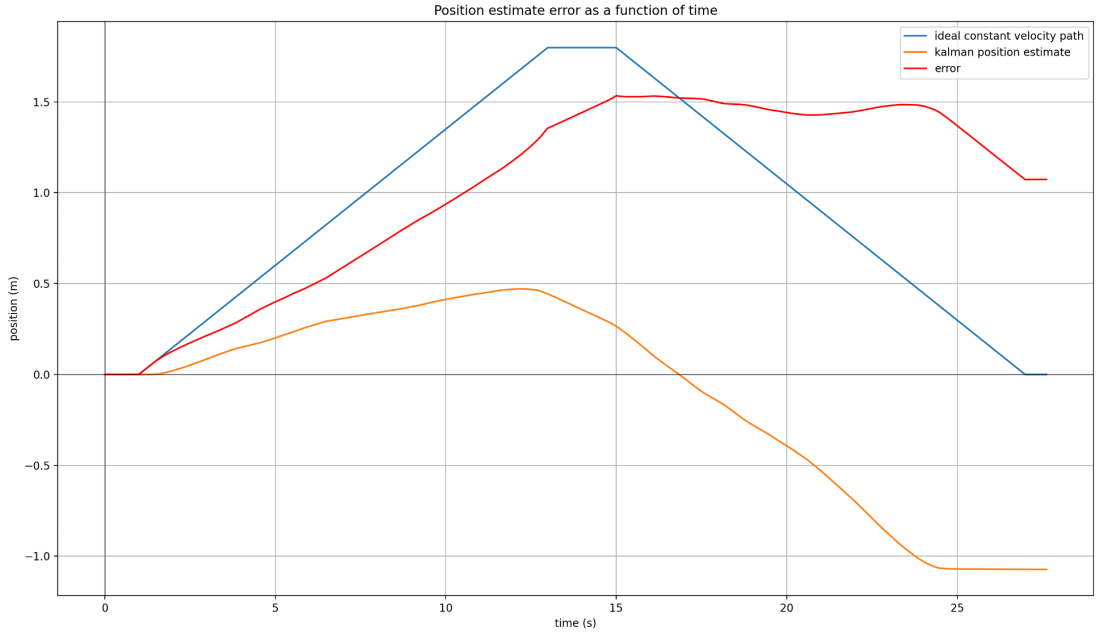


Fast vs Slow Movement

The results above are all variations on the same dataset from when the sensor was moved 6ft in the x-direction quickly. We also recorded some data where the sensor moved slower, but noticed quickly that the signal-to-noise ratio dramatically worsens, which strongly impacts the precision and accuracy of the results [Fig. 9]. Intuitively this makes sense, as higher speeds of movement results in stronger acceleration, which results in higher SNR.

For very slow motion [Fig. 10], the noise becomes nearly indistinguishable from the actual movement.

Fig 9: 6ft, slow, there-and-back (minimum-clamping on accel)



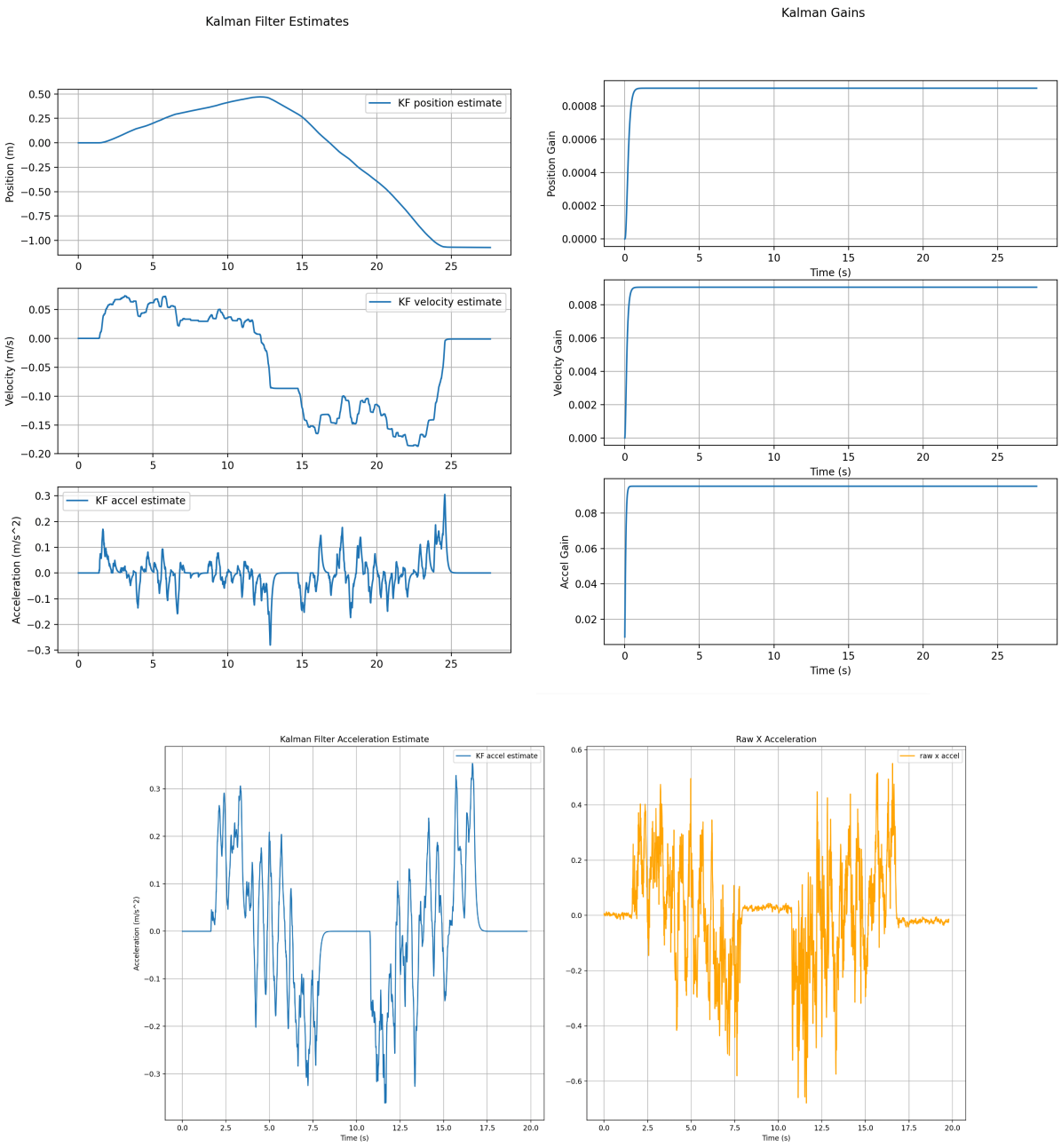
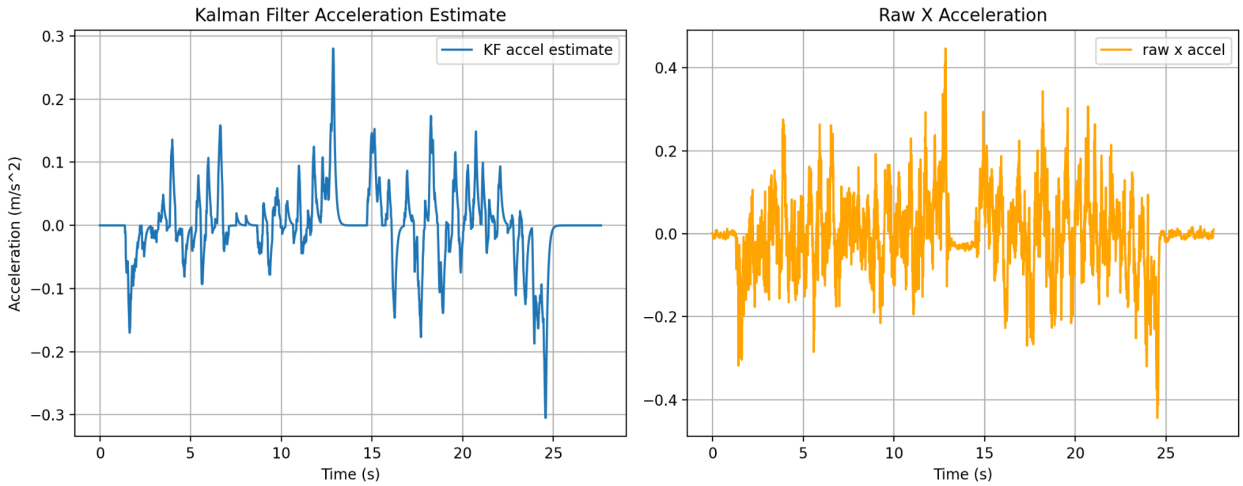


Fig. 10: Very slow movement (near stationary)



Final Approach

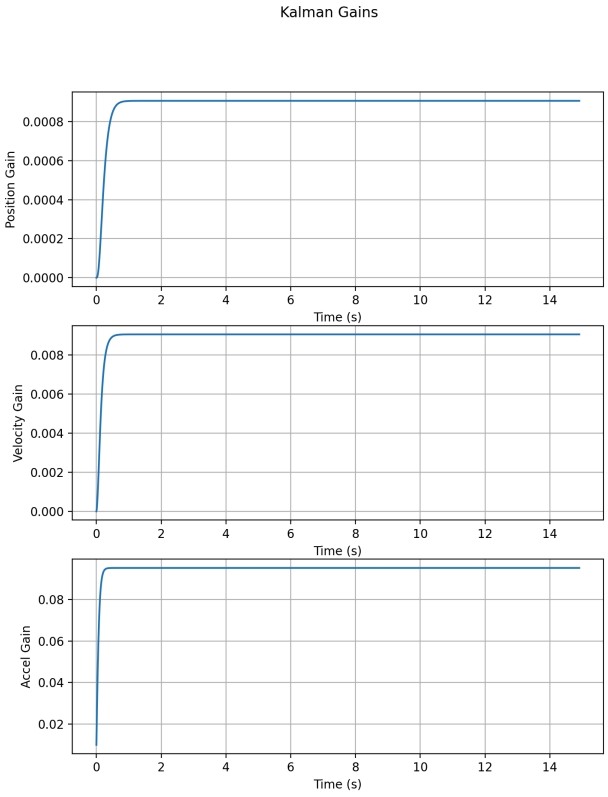
We saw the best results for minimum-clamping on accel and thus chose the approach shown in [Fig 6].

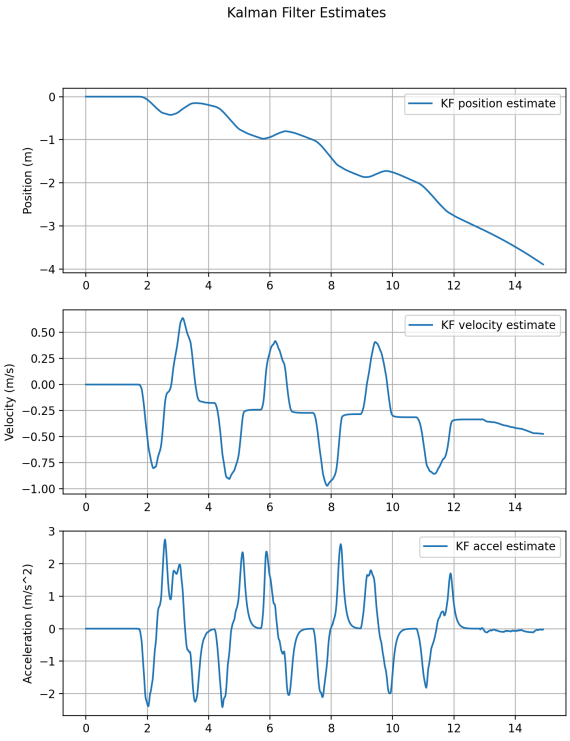
Aggregate Error

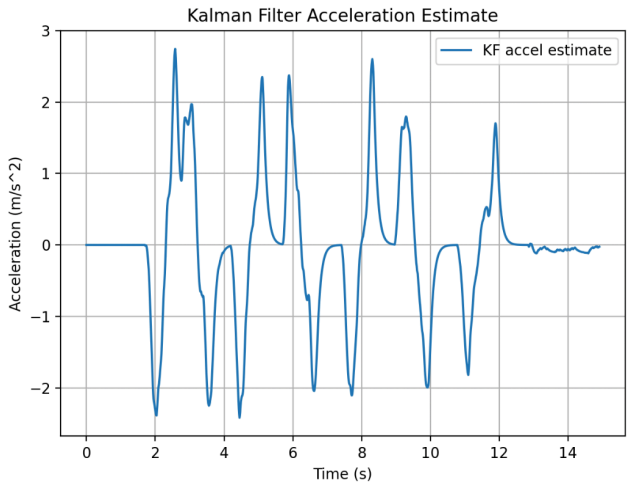
To see the aggregate error, we collected data by doing rounds of back and forth over a 1ft distance, both at a faster pace [Fig. 10] and a slower pace [Fig. 11]. Note that the direction of the drift is reversed in the two plots as we accidentally collected these measurements with different orientations in the x-axis. As before, we observed higher SNR for faster speeds and lower SNR for slower speeds from the raw data and the results.

In figure 12, we tested the approach that involves clamping. Especially at slower speeds, this stability approach may be used to keep error accumulation to a minimum as it significantly decreased our accumulated error:

Fig 10.a: 1ft, fast speed, many back-and-forth in a nearly-straight line (minimum-clamping on accel)







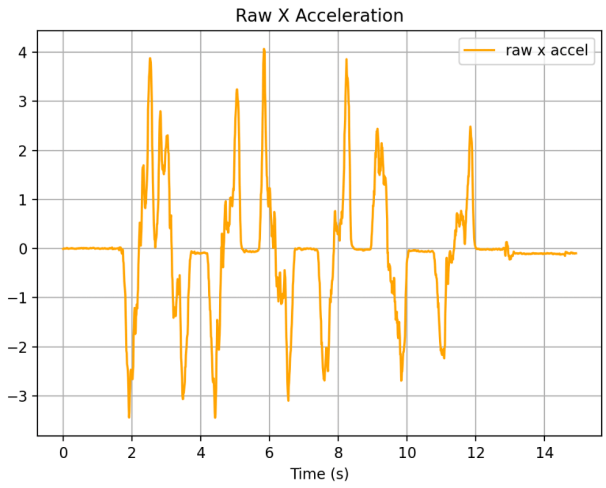


Fig 10.b: Error annotation for 1ft, fast speed, many back-and-forth in a nearly-straight line

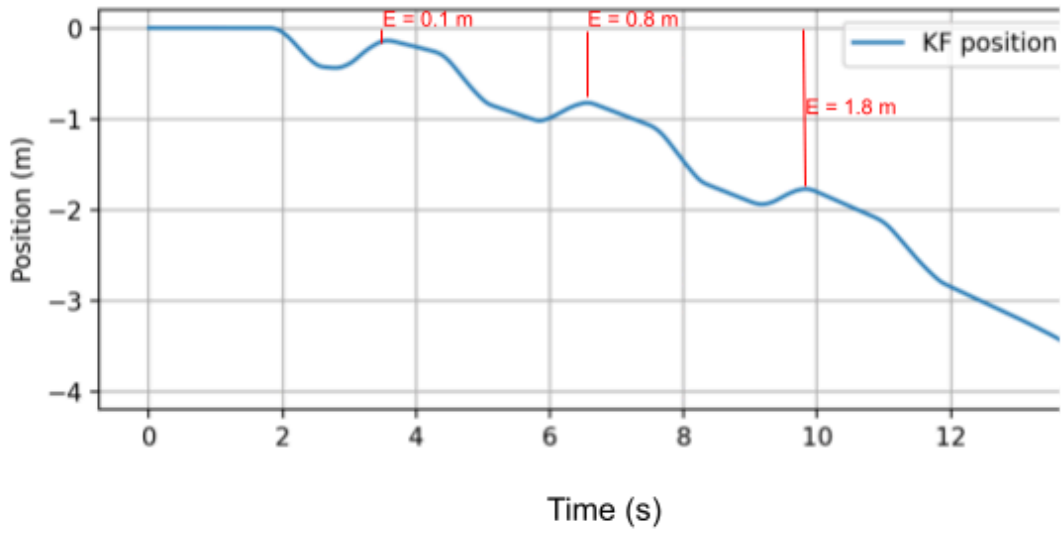
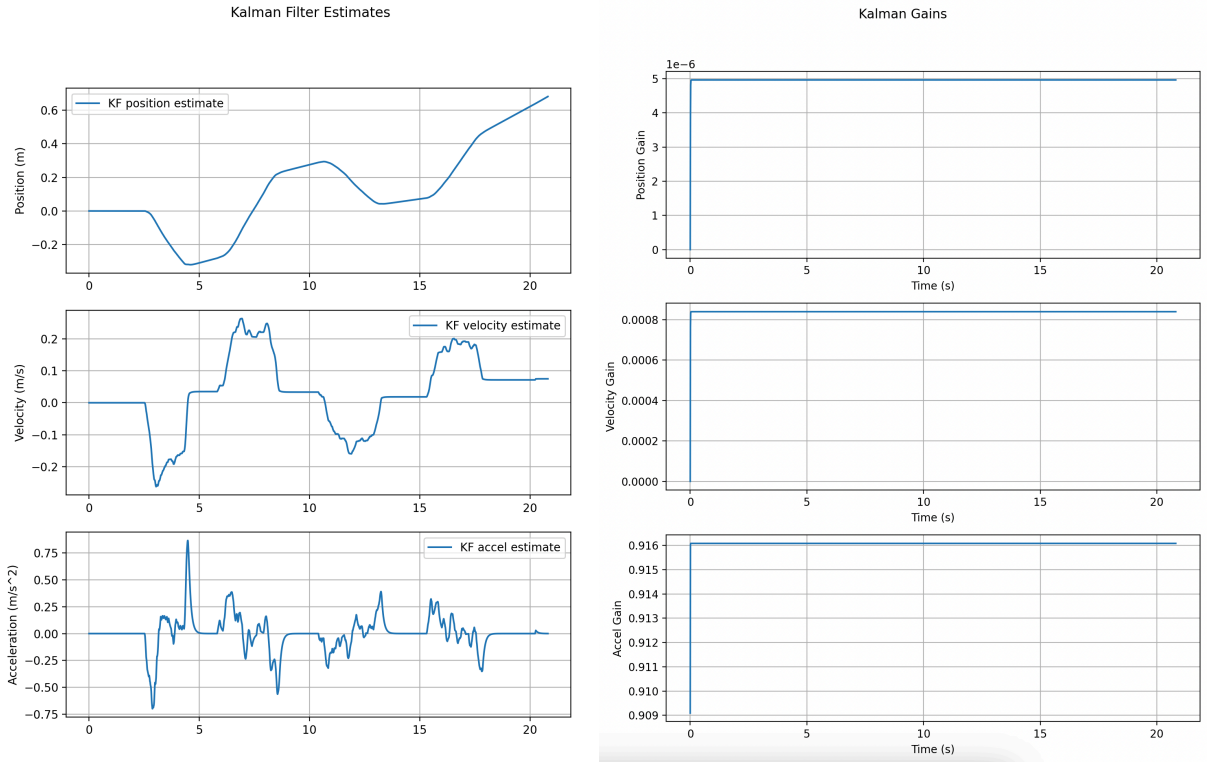


Fig 11.a.: 1ft, slow speed, many back-and-forth in a nearly-straight line (minimum-clamping on accel)



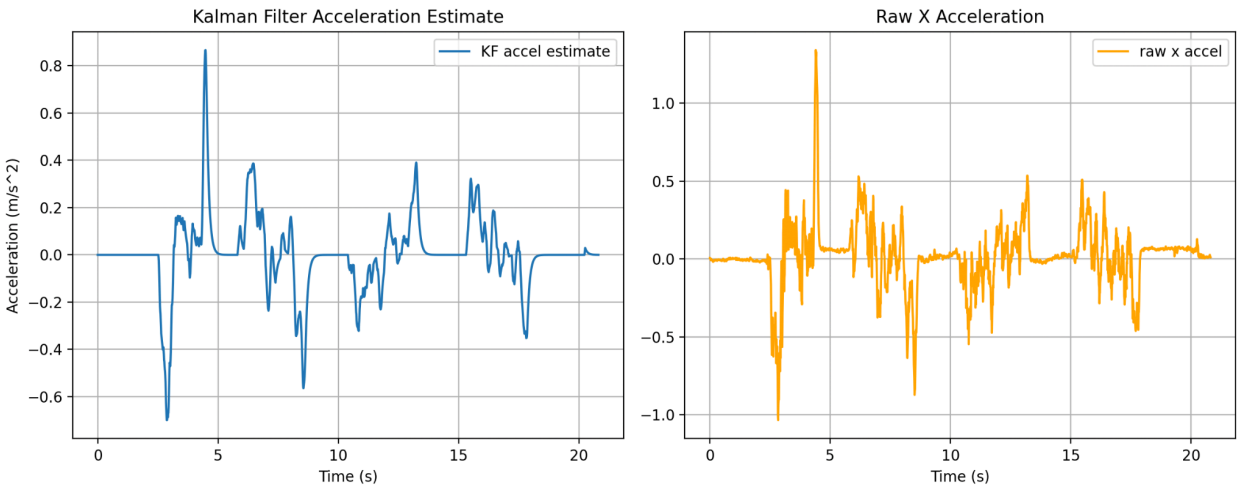


Fig 11.b: Error annotation for 1ft, slow speed, many back-and-forth in a nearly-straight line

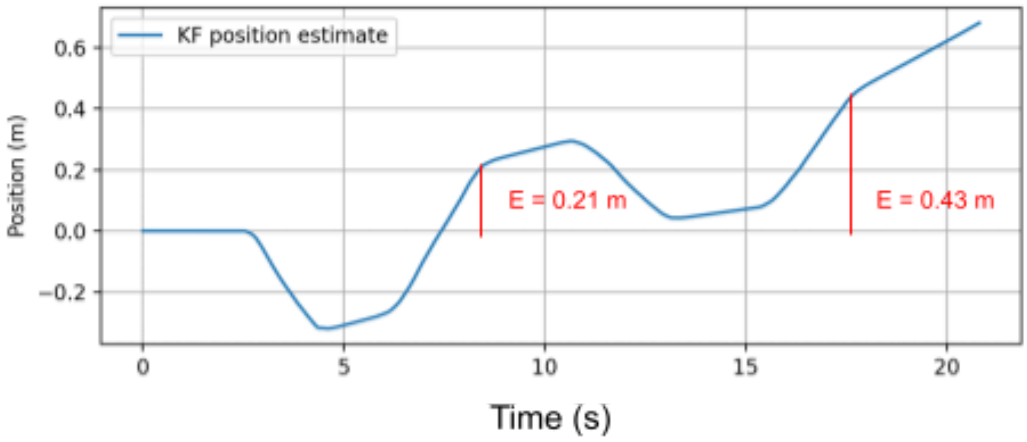
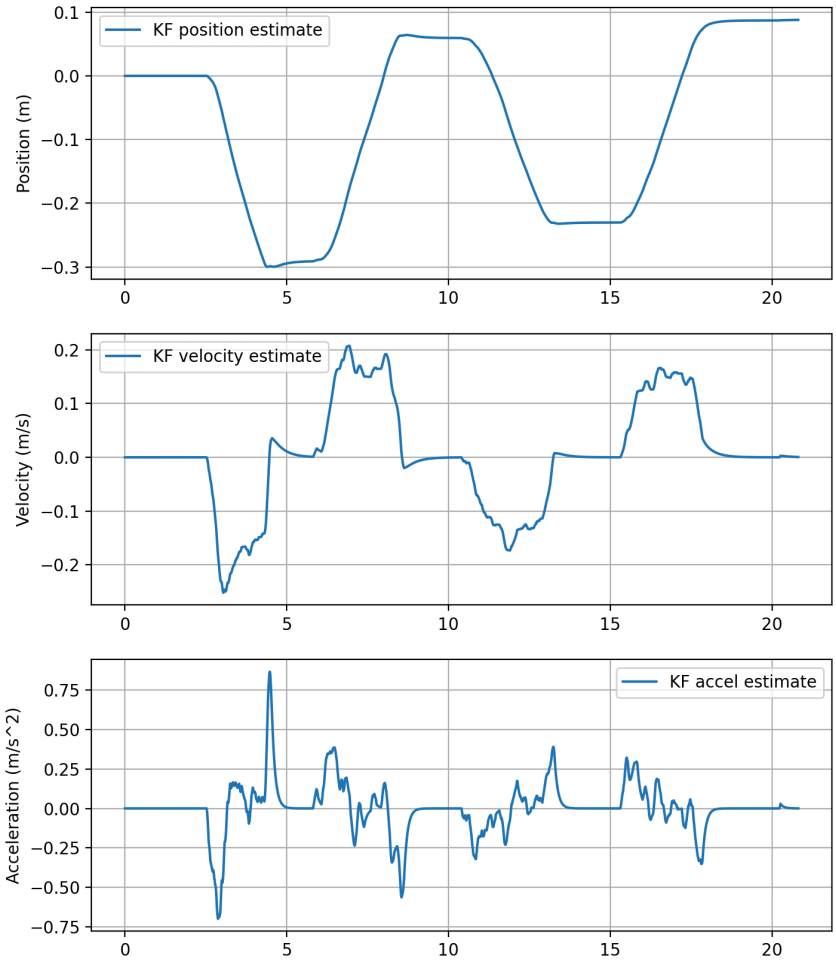
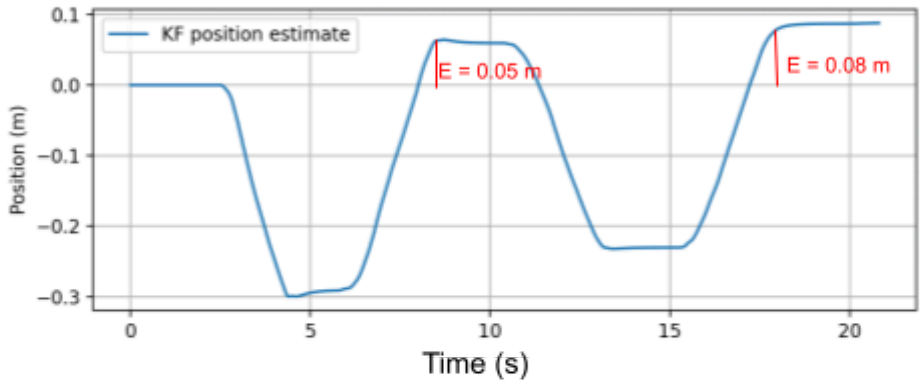


Fig 12: 1ft, slow speed, many back-and-forth in a nearly-straight line (minimum-clamping on accel and diminishing velocity bias for low velocities; this approach sacrifices accuracy for more stability).

Kalman Filter Estimates





Part 3: Implementing a 2-D Kalman filter to determine position and trajectory based on acceleration measurements in a multiple axes (X-Y plane)

1. We now create a Kalman filter that uses the X/Y axes of the accelerometer as well as the yaw sensor to determine position in an X/Y plane. We have a transfer function that includes position, velocity and acceleration for both x and y axes. At each step, we calculate the new x and y acceleration from the sensor reading and current yaw state.
2. The Kalman naturally fuses all three pieces of data (two accels (x and y) and one gyro).
3. As above, we moved our sensor in straight lines. We moved 6ft right, 6 ft up, 6ft left, and 6ft down, which should bring us to the start, but we see that starting the first direction change, we hit problems that accumulate some error.

Given our performance metric is deviation after the sensor has returned to the starting point, we have 1.35 meters of error, which is only about 18.75% of the entire distance (1.8m x 4).

On the plot, if we analyze the motion in only x axis or only y axis through [Fig. 14], we see that overall, we have a pretty good estimate. This fusion with the gyro has a worse accumulated error across the 12 feet traveled than the single-axis accelerometer because the problem is a lot more u

Fig. 13: 2D Kalman Position estimate for a 6ft by 6ft planar square motion

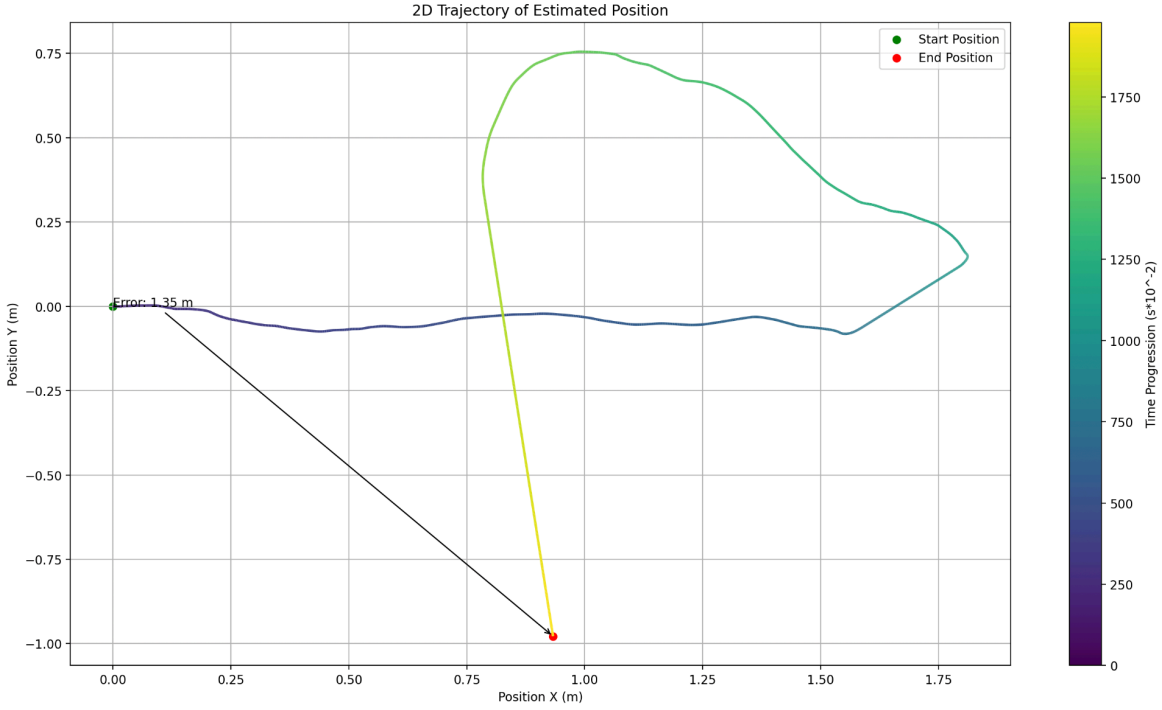

2D Trajectory of Estimated Position
Start Position
End Position
Error: 1.35 m
Position Y (m)
Position X (m)
Time Progression (s*10^-2)
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1750
1500
1250
1000
750
500
250
0

Fig. 14:

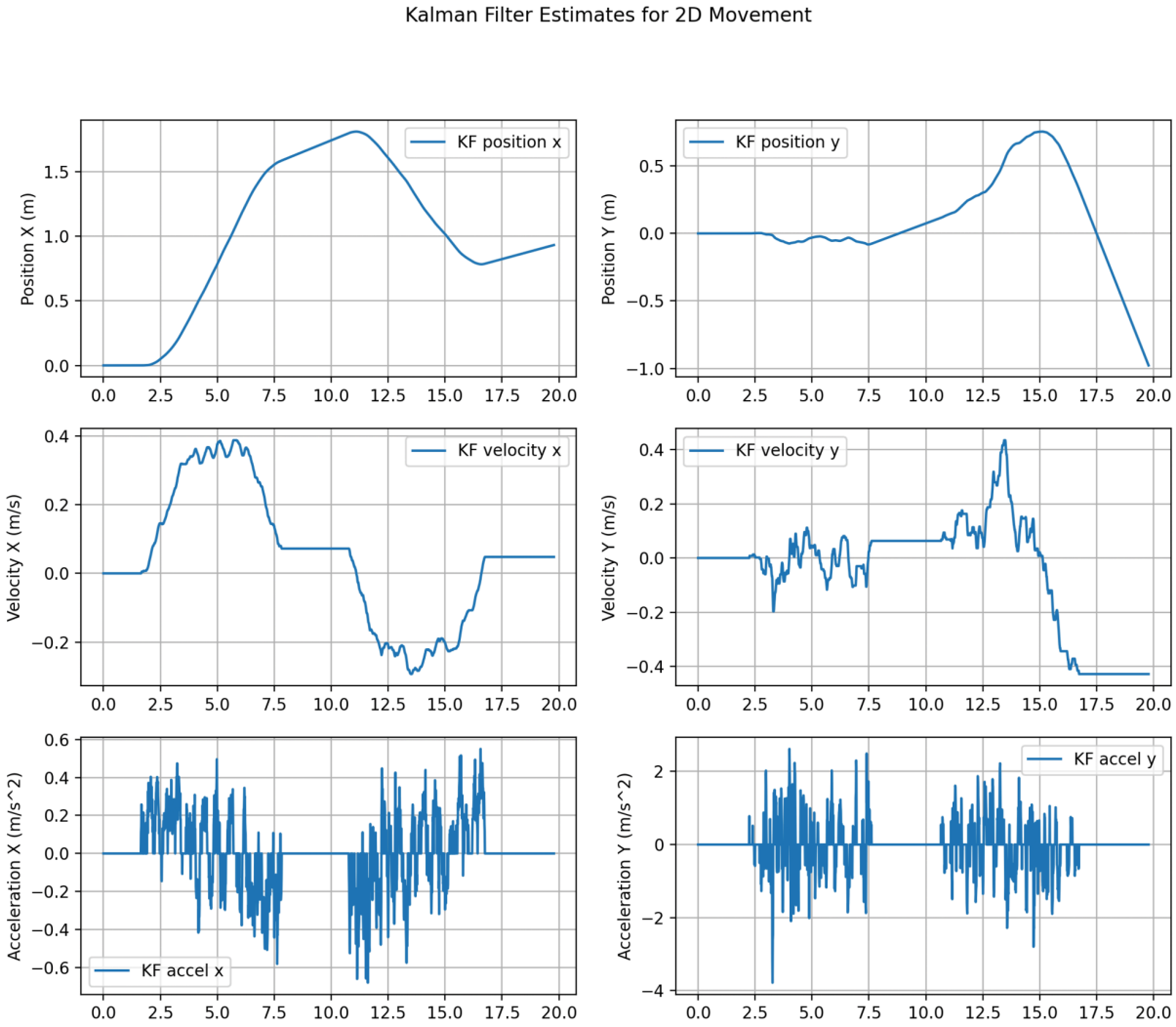
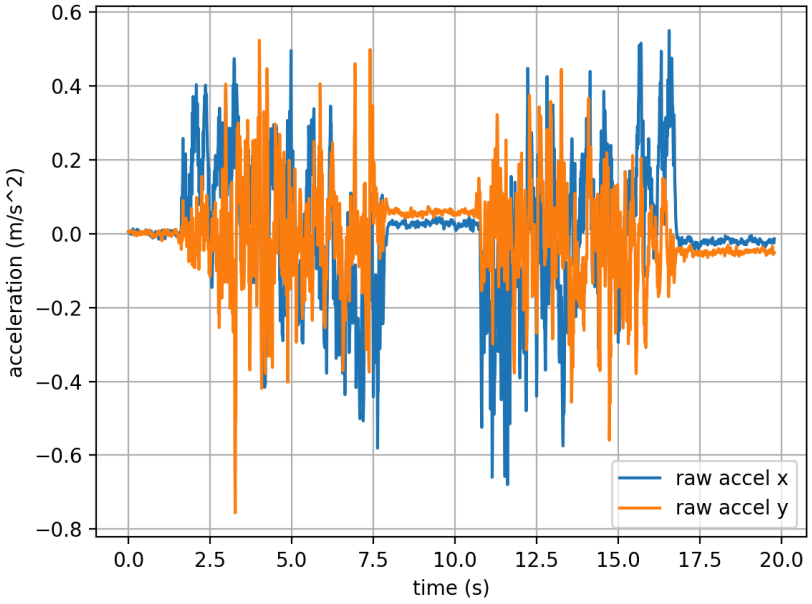

Kalman Filter Estimates for 2D Movement
KF position x
Position X (m)
KF position y
Position Y (m)
KF velocity x
Velocity X (m/s)
KF velocity y
Velocity Y (m/s)
KF accel x
Acceleration X (m/s^2)
KF accel y
Acceleration Y (m/s^2)

Fig. 15:

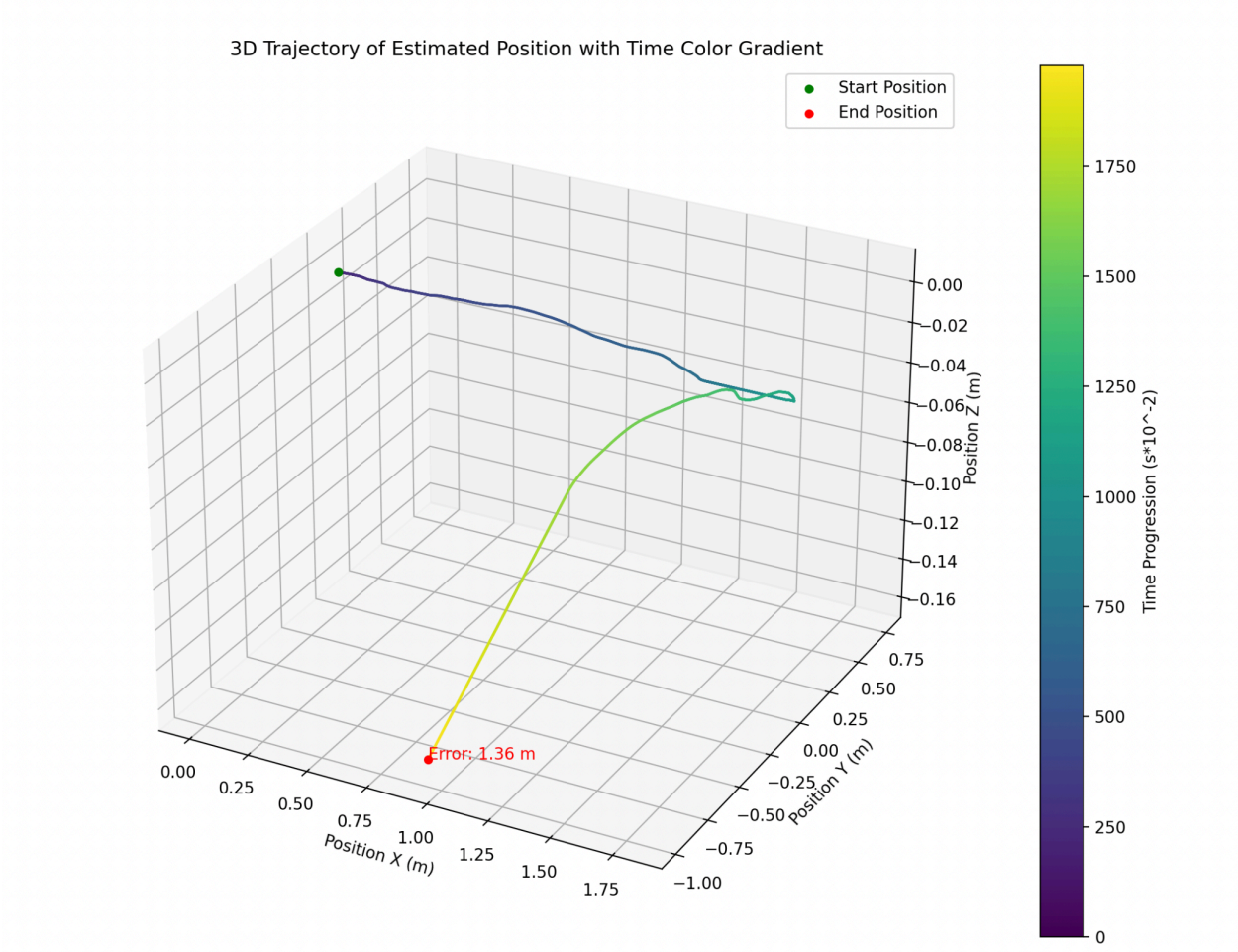


From Fig. 15, we can see that, overall, the x-axis looks a lot more symmetrical on the way there and back compared to the y-axis, which seems more asymmetrical and also seems to be the source of a lot of the error in the previous plot. One hypothesis we have is that the sensor header is more prone to movement and change of orientation on its y-direction that its x-direction, which might cause some unintentional tilt, movement, and error accumulation on the y-axis.

Part 4: Implementing a 3-D Kalman filter to determine position and trajectory based on acceleration measurements in all axes (X-Y-Z space)

Using the same raw data as Part 2 (6ft by 6ft square, as planar as we could movement), we also wanted to see the leakage and error on the z-axis.

Fig. 16: 3D Kalman Position estimate for a 6ft by 6ft planar square motion



We see the leakage and drift on the z-axis although there was minimal movement on that axis.

We see that some of the error and motion accumulates on the z-axis, which was hidden on our plot for Part 3.

References used (read through) for Allan deviation analysis:

https://www.researchgate.net/profile/Tayfun-Akin/publication/224351551/figure/fig10/AS:399668698730505@1472299851625/Allan-variance-plot-of-the-fabricated-gyroscope-obtained-by-processing-the-measured.png

https://miro.medium.com/v2/resize:fit:1400/1*DNQ_HPfVew15DPIfqU_0JQ.png

https://mwrona.com/posts/gyro-noise-analysis/

https://www.phidgets.com/docs/Allan_Deviation_Guide

https://edstem.org/us/courses/66163/discussion/5476742

https://edstem.org/us/courses/66163/discussion/5486378

https://www.phidgets.com/docs/Allan_Deviation_Guide