AUV Localization Using Dead Reckoning Techniques with Integrated Sensors

Ryan Gochar, *MS student,* ***Brandon Davis****, MS student, University of Florida Department of Electrical and Computer Engineering*

Abstract*—As autonomous vehicles become more widely desired for solving complex problem sets, the need for more robust navigation solutions will increase. Most advanced behaviors we ask autonomous vehicles to complete require a robust vehicle position estimate or localization solution. For vehicles primarily being used outdoors, using the global navigation satellite system (GNSS) is currently the most robust positioning system available for use. Increasingly autonomous systems are being used in environments where GNSS is not a suitable positioning solution. The most increasingly common are areas where GNSS is not available at all such as underwater when using autonomous underwater vehicles (AUV). For these use cases alternate methods of creating position estimates are needed for the vehicle to know its location. In this project, using an IMU sensor to get orientation and velocity data and using a dead reckoning algorithm to estimate position is explored. Using two IMU sensors, a high-precision and low-precision sensor, a Kalman Filter was used to filter the raw data from the IMUs, and position estimates were estimated for a single path with various GNSS update frequencies and simulated Doppler Velocity Log (DVL) update frequencies. Finding that the RMSE error between IMU sensors was not appreciable. Further findings include no DVL updates performed better than infrequent DVL updates, considered less than 25% of the task duration, but frequent DVL updates had the best performance. In all cases, RMSE error decreased as the GNSS update frequency increased. **Keywords—Dead Reckoning, IMU, DVL**

I. Introduction

The purpose of this project is to implement a dead reckoning localization solution on an autonomous underwater vehicle (AUV). Autonomous vehicles require robust position estimates and orientation data to achieve set out tasks. AUVs are becoming increasingly popular due to the wide range of uses and the ability to conduct long missions without a human body onboard in control. Some of the missions that AUVs are being used for are for scientific exploration and data collecting, such as ocean mapping or searching for oil or gas deposits for potential oil rig sites. AUVs are also being used for military applications like reconnaissance missions. On surface vessels, traditionally a global positioning satellite system (GPS) is used to receive an absolute position. However, water has a very high attenuation effect on radio waves, making it near impossible to receive any accurate GPS updates on an AUV due to the depth of water they are typically operating in. There is an increasing need for alternative methods of localization in environments where it is not feasible to use GPS.

One method of determining position for an AUV which may not have access to frequent GPS updates is dead reckoning. Dead reckoning is the process of estimating the position of an object using a known starting point, orientation, velocity, and elapsed time to calculate a position estimate. In autonomous vehicles, a common technique for dead reckoning is the use of an Internal Measurement Unit (IMU). An IMU is typically a small device which can be placed onboard an AUV which provides real time data about the movement which it experiences. For this project, a 9 degree of freedom IMU was simulated. The 9 degree of freedom IMU has a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer. The accelerometer measures the linear acceleration of the IMU, the gyroscope measures the angular velocity around these axes, and the magnetometer provides the strength and direction of the magnetic field which can be useful in determining headings. The combination of these 3 sensors gives the data required to conduct dead reckoning algorithms to estimate the position of an AUV.

This project also explores the integration of an additional sensor typically found on an AUV, the doppler velocity log (DVL). A DVL is an acoustic sensor which makes it viable for use in underwater applications. The DVL works by emitting acoustic pulses from different transduces positioned along the body of the vehicle and measuring the Doppler shift upon receiving an echo of the pulse. DVL's then use this to calculate the velocity of the vehicle relative to the sea floor. Since the DVL relies on an echo to measure velocity, it is most accurate at depths close to the sea floor, or any surface of whatever environment the AUV is in. The measured velocity can be input into the dead reckoning algorithm, and this study looks at the relation of the frequency of DVL updates relative to the duration of the task and the accuracy of the dead reckoning position estimates.

A study conducted in 2010, "Accuracy Analysis of DVL/IMU/Magnetometer Integrated Navigation System using Different IMUs in AUV" set out to explore the difference in performance of different qualities of IMUs in position estimates through dead reckoning with the integration of DVL and magnetometer recordings [4]. This study found that performance was indistinguishable from the different IMU sensors, proposing that lower quality IMU sensors were just as effective for dead reckoning when integrated with DVL and magnetometer fusion. This study makes use of a Kalman filter for the IMU data. Another study, "Improved Dead Reckoning Localization using IMU sensor", proposes a very nice method for a dead reckoning algorithm,

showing accurate results with no velocity updates [2]. This paper aims to use the principles found in previous studies to further examine the effects of sensor integration with dead reckoning algorithm, implementing the widely used Kalman filter on the raw data from the IMU. The paper is outlined as:

II. Methodology

A. Dataset

The dataset for this project was simulated using an open-source software which allows the input of IMU specifications and path instructions to generate simulated IMU and GPS data. The IMU specifications were tailored to the Bosch BNO055, a 9 degree of freedom MEMS IMU, and the generated path for the simulation was a square path with a tight loop at one corner, as seen below in Figure 1.

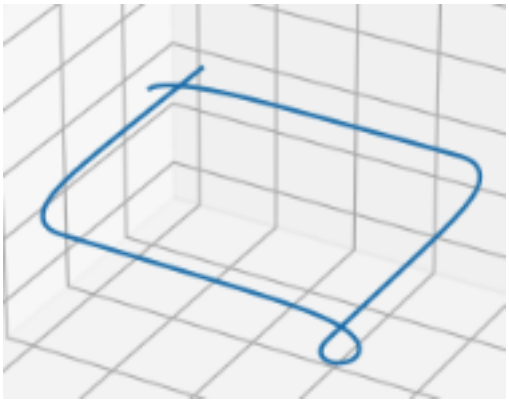


Figure 1: Simulated Target Path

One of the goals of this project was to explore and verify findings from previous studies which suggest that lower quality IMUs can produce similar results as higher quality IMUs in dead reckoning applications. The way that this project aims to study this is by inserting noise into the acceleration data from the simulated IMU. Noise was added to the acceleration data at two values, which were functions of the amplitude of the acceleration. The first, simulating a higher precision IMU, was given noise uniformly distributed with amplitude of 10% of the amplitude of the acceleration. The second, lower precision IMU was given noise at 50% its amplitude.

B. Open Loop Dead Reckoning Solution

1) Kalman Filter

IMU data is inherently very noisy, which can be a problem when precise location estimates are trying to be estimated. One widely used method of filtering IMU data is the Kalman filter [3]. The Kalman filter, developed by R.E. Kalman in a paper "A New Approach to Linear Filtering and Prediction Problems" is a widely used linear estimator for time varying signals, like an EEG signal [8]. The principle of a Kalman filter is to make an estimate of a system state based on previous data, then record a new sample and measure how close the sample is to the estimate. If the sample is far away from the estimate, the Kalman gain will diminish, and you will prefer the estimate. If the sample is close to the estimate, the filter will prefer the sample.

Kalman Filter Algorithm

Initial Conditions :

$$\hat{x}(0|0), \quad \tilde{P}(0|0)$$

Prediction :

$$\hat{x}(t|t-1) = A\hat{x}(t-1|t-1) + Bu(t-1)$$

$$\tilde{P}(t|t-1) = A\tilde{P}(t-1|t-1)A + R_{ww}(t-1)$$

Innovation :

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - C\hat{x}(t|t-1)$$

$$R_{ee}(t) = C\tilde{P}(t|t-1)C^T + R_{vv}(t)$$

Gain :

$$K(t) = \tilde{P}(t|t-1)C^T R_{ee}^{-1}(t)$$

Update :

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t)$$

$$\tilde{P}(t|t) = \left[I - K(t)C\right]\tilde{P}(t|t-1)$$

Figure 2: Kalman Filter Algorithm

Above is the Kalman filter algorithm [1], where $\hat{x}(0|0)$ is the initial state estimate, $\tilde{P}(0|0)$ is the initial error covariance matrix, $\hat{x}(t|t-1)$ is the predicted state estimate at time t given observations up to $t-1$. $\tilde{P}(t|t-1)$ is the error covariance estimate at time t given observations up to $t-1$. $y(t)$ is the measurement, $K(t)$ is the Kalman gain, and the outputs are the updates to the state estimate and the error covariance. After the initial conditions are established, the algorithm is run for every time step t.

2) Calculation of Position

Next an open loop dead reckoning approach was implemented, with the goal of estimating position and verifying against the ground truth GPS positions. Position estimates can be obtained by integrating the acceleration retrieve velocity estimate, then again integrating velocity over a time interval to obtain position. This algorithm was very simply implemented through these basic principles.

$$v[k] = v[k-1] + a * dt \quad (1)$$

$$p[k] = p[k-1] + v * dt \quad (2)$$

Where at each iteration k, where $v[k-1]$ and $p[k-1]$ are the previous estimates, initialized at 0, and dt is the time interval at which the data is collected. These equations were implemented in python as follows:

```python
#Calculate Orientation

def calculate_orientation(a_x, a_y, a_z, previous_pitch, previous_roll):

    alpha = 0.9999

    theta_z = np.arctan2(a_y, a_z)
    theta_x = np.arctan2(a_x, np.sqrt(a_y**2 + a_z**2))
    pitch = (previous_pitch * alpha) + (theta_x * (1 - alpha))
    roll = (previous_roll * alpha) + (theta_z * (1 - alpha))

    return pitch, roll
```

```python
#Calculate Velocity

def calculate_velocity(a_x, a_y, a_z, previous_velocity_x, previous_velocity_

    velocity_x = previous_velocity_x + a_x * delta_t
    velocity_y = previous_velocity_y + a_y * delta_t
    velocity_z = previous_velocity_z + a_z * delta_t

    return velocity_x, velocity_y, velocity_z
```

```python
#Calculate Position

def calculate_position(previous_position_x, previous_position_y, velocity_x,

    position_x = previous_position_x + velocity_x * delta_t
    position_y = previous_position_y + velocity_y * delta_t

    return position_x, position_y
```

Figure 3: Python Implementation of DR Algorithm

3) Integration of GPS Ppostion Updates

Next, GPS positions were integrated at various frequencies. First, just providing a single GPS update, to simulate an AUV which rises near the surface enough to get a GPS update. Then, more regularly to simulate an AUV working near the surface or can receive true position signals another way, such as acoustically. The algorithm integrates GPS data into the calculation of position, which searches for instances when GPS updates are available, collects the GPS update and updates the previous position with the known position from the GPS.

4) Integration of DVL Updates

Finally, this open loop dead reckoning algorithm integrates velocity updates, simulating an AUV with a DVL onboard which can calculate velocity. For this project, since the data generation software used did not include DVL options, the velocity data from the GPS was used as a simulated DVL. The DVL updates were implemented in the exact same manner as the GPS updates, where if a DVL update was available, it would update the previous velocity to be used in the next iteration with the true velocity.

C. Error-state Kalman Filter Dead Reckoning

In addition to the open-loop Kalman integration from the previous section we explore an indirect feedback Kalman filter approach based on Fossen's Handbook [5]. This filter defines a full 15-state *Inertial Navigation System (INS)* and is how it will be referred to. An INS defines an entire navigation suite of sensors usually comprised of IMU, GNSS, and some velocity source [5]. For the feedback filter error estimates are used to update the state estimates [5]. This reduces the overall drift of the solution by keeping the cumulative error low. Figure 4 shows the overall architecture of the feedback Kalman filter inertial based navigation solution.

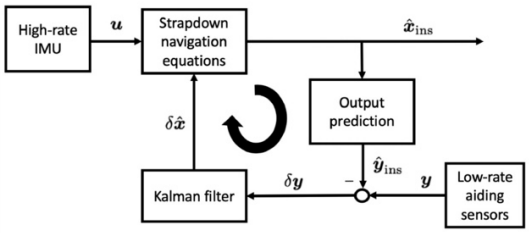


Figure 4: Feedback filter for inertial navigation system [2]

Very similar to the first implementation, this solution can be aided by periodic updates from additional sensors. These updates can come from many different sensor sources but most commonly for AUVs come from GNSS or DVL. For this implementation we are simulating a lower frequency external velocity measurement and low frequency position measurements.

The general approach for this algorithm begins with a high-level understanding of the equations involved. There are two mode components of this algorithm: slow measurements (low frequency) and fast measurements (high frequency). After each low frequency measurement, the 15-state vector $\hat{x}_{ins}$ is updated and the error-state vector goes to zero [5]. The mathematical representation follows equation 3 [5].

$$\hat{x}_{ins}[k] \leftarrow \hat{x}_{ins}[k] + \delta\hat{x}[k] \tag{3}$$

$$\delta\hat{x}[k] = [\delta\hat{p}^{n}_{nmI}[k]^T, \delta\hat{v}^{n}_{nmI}[k]^T, \delta\hat{b}^{b}_{acc}[k]^T, \delta\hat{\Theta}_{nb}[k]^T, \delta\hat{b}^{b}_{ars}[k]^T]^T \tag{4}$$

$$\hat{x}_{ins}[k] = [\hat{p}^{n}_{ins}[k]^T, \hat{v}^{n}_{ins}[k]^T, \hat{b}^{b}_{acc,ins}[k]^T, \hat{\Theta}_{ins}[k]^T, \hat{b}^{b}_{ars,ins}[k]^T]^T \tag{5}$$

The error-state vector is defined in equation 4 and 15-state vector of AUV defined by equation 5 [5]. The 15-state vector contains all the current estimated states of the system: *position, velocity, bias,* and *orientation*. These states are states measured in the *INS* frame *{ins}* with respect to the *North-East-Down (NED)* frame *{n}*. For this project we are assuming that the *INS* and *Base* frame *{b}* of a vehicle have the same origin and are aligned for a strapdown navigation system. The strapdown navigation system means that all sensors are rigid mounted to a vehicle body and therefore are measuring the movements of vehicle body. The strapdown navigation equations in equations 6-8 are the basis of this feedback Kalman filter [5].

$$\dot{p}^{n}_{nmI} = v^{n}_{nmI} \tag{6}$$

$$\dot{v}^{n}_{nmI} = a^{n}_{nmI} \tag{7}$$

$$\dot{\Theta}_{nb} = T^{n}_{b}(t)\omega^{b}_{nb} \tag{8}$$

The strapdown navigation equations define the relationship between the measurements from inertial sensors and vehicle states *position, velocity,* and *orientation*.

Once the INS states are defined the Kalman filter makes predictions $\hat{P}^{-}[k+1]$. Corrections are made using the feedback architecture and state estimates propagate based on equations 9-11 [5].

$$\hat{p}^{n}_{ins}[k+1] = \hat{p}^{n}_{ins}[k] + h\hat{v}^{n}_{ins}[k] \quad (9)$$

$$\hat{v}^{n}_{ins}[k+1] = \hat{v}^{n}_{ins}[k] + h(R^{b}_{n}[k](f^{b}_{imu}[k] - \hat{b}^{b}_{acc,ins}[k]) + g^{n}) \quad (10)$$

$$\hat{\Theta}_{ins}[k+1] = \hat{\Theta}_{ins}[k] + hT^{b}_{n}[k](\omega^{b}_{imu}[k] - \hat{b}^{b}_{ars,ins}[k]) \quad (11)$$

To implement this feedback Kalman filter MATLAB was used with helper functions from the *MSS* library [5]. The MSS library also contains other various sensor data processing tools as well as sensor simulation.

III. Results

A. Open Loop Dead Reckoning Solution

Dead reckoning position estimates were determined for various iterations DVL and GPS update frequencies and were shown graphically over the GPS position which was used as the ground truth. The frequencies at which the GPS and DVL updates were implemented were based on a % of the full path, where 50% would lead to a single midpoint update, 10% would lead to updates every 10% of the path. The following update frequencies were implemented with the subsequent notation: 100% - no update, 50% - midpoint update, 25% - infrequent update, 10% - frequent update, 5% - very frequent update, and 1% - near continuous update.

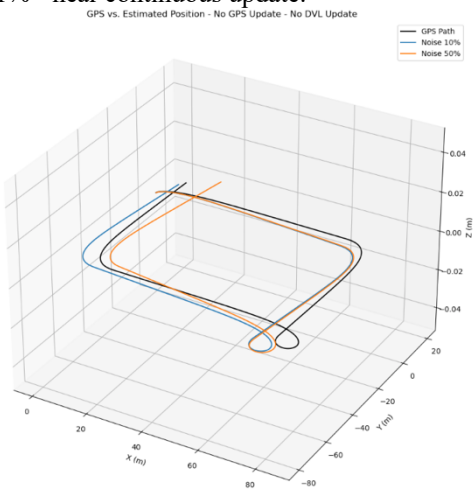


Figure 5: GPS vs. Estimated Position - No GPS Update - No DVL Update

In Figure 5., there are no GPS or DVL updates, showing the raw output of the dead reckoning algorithm for the simulated high and low precision IMUs which were represented by introducing uniformly distributed noise at 10% (in blue) and 50% (in orange) of the amplitude of the raw acceleration data. Next, five more runs with increasing frequency of DVL updates, still with no GPS updates were conducted and plotted against the true GPS position seen in black. The progression of DVL update frequency can be seen in Appendix I, with the final estimate with no GPS update and near continuous DVL updates seen below in Figure 6.

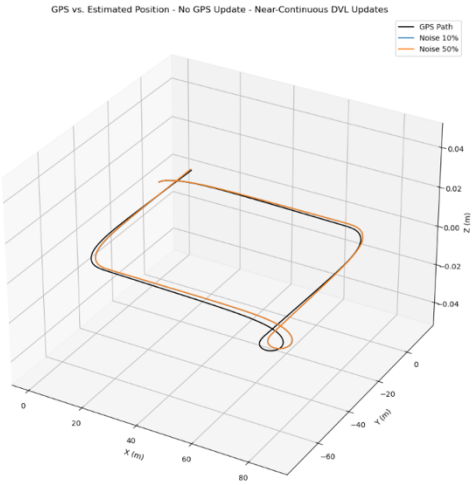


Figure 6: GPS vs. Estimated Position - No GPS Update - Near Continuous DVL Update

This process was repeated over a range of GPS update frequencies, implemented in the same manner as the DVL updates. This process would simulate if the AUV was operating near the surface and able to receive accurate GPS updates, or for dead reckoning applications to an autonomous surface vessel (ASV). The complete results for these experiments can be found in Appendices I-VI. The result of the final DR estimate, with near continuous GPS and near continuous DVL updates is seen below in Figure 7.

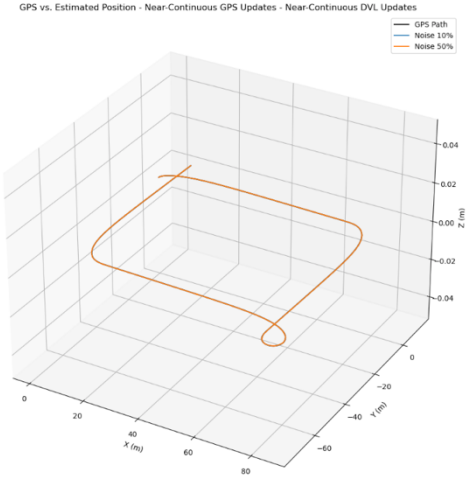


Figure 7: GPS vs. Estimated Position - Near Continuous Update - Near Continuous DVL Update

B. Error-state Kalman Filter Dead Reckoning

Using simulated sensor data position estimates were made using the error-state Kalman filter. Comparisons were made both visually by plotting and computing error. The estimated positions were plotted over the ground truth GPS data for a visual comparison of true path vs. dead reckoned path. The simulated sensor data for this portion of the experiment contained 3-DOF data. An example of the simulated path is in Figure 8. A systematic approach was taken for varying the sensor parameters and update rates. Figure 8 shows the error-state filter aided with a 10% GPS update rate. For every

variation in GPS update rate, the filter was run with low frequency velocity data input and without. Figure 9 shows the same 10% GPS update rate but aided with low frequency velocity input at the same 10% rate.

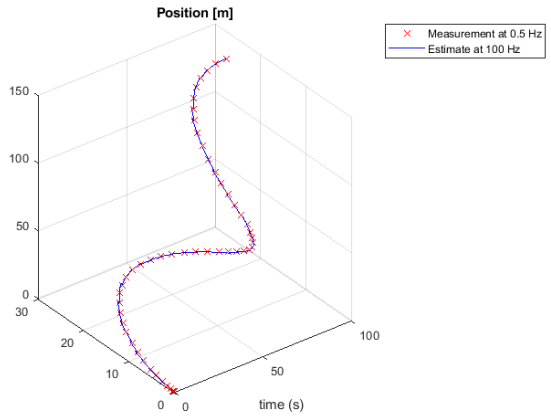


Figure 8: GPS vs. Estimated Position – 0.5 Hz GPS Update – No Velocity Aiding

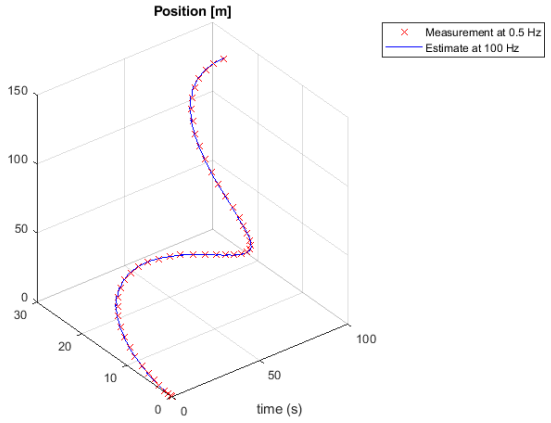


Figure 9: Estimated Position – 0.5 Hz GPS Update –0.5 Hz Velocity Aiding

This same process was repeated for a range of different GPS update frequencies, each with and without velocity aiding. Additional results for the various parameter changes are found in Appendices VII.

IV. Analysis

A. Open Loop Dead Reckoning Solution

The outcomes of the dead reckoning algorithm were then analyzed for accuracy using a root mean squared error (RMSE) approach. The RMSE measures the difference between the ground truth position and the DR estimate. The RMSE was analyzed in two ways. First, as an average distance from the ground truth, measured in meters. Second, the RMSE was divided by the cumulative distance traveled to the point of measurement, giving the RMSE as a percentage of the distance traveled. The first criteria that was examined was the overall performance of both simulated IMU sensors.

In Figure 10., the high precision IMU only appreciably outperformed the low precision IMU when no GPS and no DVL updates were present. However, the low precision IMU and high precision IMU perform almost identically throughout the variations of the trials, with no clear separation of the two to show that one would be better than the other for this task.

Average RMSE: per distance traveled (%) - No GPS Update						
DVL Update Freq:	No update	50%	25%	10%	5%	1%
High Precision:	**1.53%**	**2.26%**	**2.52%**	**1.13%**	**0.79%**	**0.60%**
Low Precision IMU:	**2.83%**	**2.55%**	**2.08%**	**1.07%**	**0.95%**	**0.63%**
Average RMSE: (%) - Midpoint GPS update						
DVL Update Freq:	No update	50%	25%	10%	5%	1%
High Precision:	**1.71%**	**3.67%**	**2.46%**	**0.79%**	**0.61%**	**0.55%**
Low Precision IMU:	**1.60%**	**3.03%**	**2.89%**	**0.85%**	**0.67%**	**0.55%**
Average RMSE: (%)- Infrequent GPS Update						
DVL Update Freq:	No update	50%	25%	10%	5%	1%
High Precision:	**1.02%**	**1.89%**	**1.78%**	**0.72%**	**0.40%**	**0.32%**
Low Precision IMU:	**1.19%**	**2.13%**	**1.79%**	**0.66%**	**0.39%**	**0.32%**
Average RMSE: (%) - Frequent GPS Update						
DVL Update Freq:	No update	50%	25%	10%	5%	1%
High Precision:	**0.43%**	**0.75%**	**0.73%**	**0.34%**	**0.23%**	**0.17%**
Low Precision IMU:	**0.47%**	**0.73%**	**0.71%**	**0.32%**	**0.23%**	**0.17%**
Average RMSE: (%) - Very Frequent GPS Update						
DVL Update Freq:	No update	50%	25%	10%	5%	1%
High Precision:	**0.25%**	**0.39%**	**0.38%**	**0.20%**	**0.15%**	**0.10%**
Low Precision IMU:	**0.23%**	**0.38%**	**0.37%**	**0.19%**	**0.15%**	**0.10%**
Average RMSE: (%) - Near Continuous GPS Updates						
DVL Update Freq:	No update	50%	25%	10%	5%	1%
High Precision:	**0.07%**	**0.09%**	**0.09%**	**0.07%**	**0.06%**	**0.05%**
Low Precision IMU:	**0.08%**	**0.10%**	**0.09%**	**0.06%**	**0.06%**	**0.05%**

Figure 10: Average RMSE for Various GPS and DVL Update Frequencies

Figure 11. shows the error for each iteration, normalized onto the same scale. This plot shows the error for each DVL update frequency and GPS update frequency.

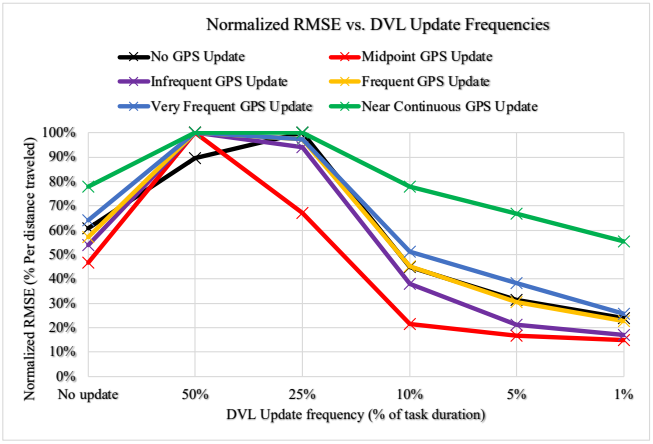


Figure 11: Normalized RMSE vs. DVL Update Frequencies

No matter the GPS update frequency, the error has a peak where DVL updates are infrequent, at both the singular midpoint update and the update rate of 25% of the trial length. These results show that the DR algorithm is consistently more accurate with no DVL updates, rather than infrequent DVL updates. It should be noted that the error with near continuous GPS updates, represented in green, had a consistently lower absolute error, ranging from 0.05% to 0.09%, which is why when normalized generally has less variance than the other GPS update rates, but the trend is still consistent where the highest error was for a midpoint DVL update and a 25% DVL update, the peak is just not as exaggerated due to the small values. Finally, the error drops significantly once frequent DVL updates are present. The error continues to decrease with more frequent DVL updates. In all cases of GPS updates, the lowest error was with the most frequent DVL updates.

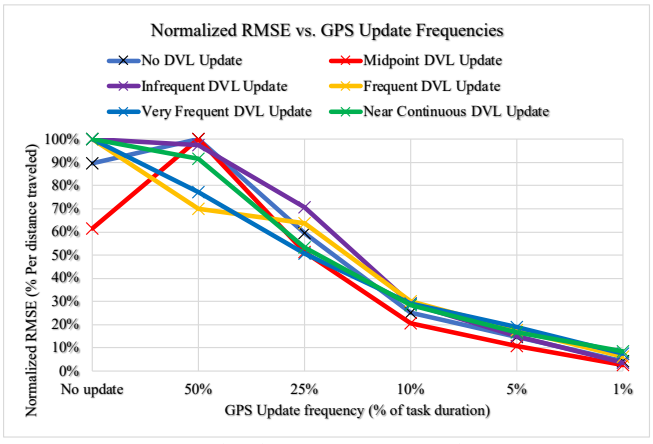


Figure 12: Normalized RMSE vs. GPS Update Frequencies

Finally, the analysis was repeated for GPS update frequencies. There is a clear relation to the error decreasing as the GPS update frequency increases. The only clear outlier to this is during the midpoint DVL update, where the combination of a single midpoint GPS update along with a single midpoint DVL update has the highest error of all the experiments.

B. Error-state Kalman Filter Dead Reckoning

The results from combinations of update rates and velocity input were calculated for error. Table 1 shows the normalized error as a percent of distance traveled. This is a common metric for INS and other guidance, navigation, and control (GNC) systems.

Error (% distance travelled)				
GPS Update Freq:	0.5 Hz	0.1 Hz	0.008 Hz	0.005 Hz
No velocity aiding	0.19%	3.88%	3.58%	12.98%
Velocity aided	0.21%	2.76%	2.25%	5.99%

Table 1: Average position error of error-state Kalman filter

The first observation made is the trend that the less frequent the GPS update the more navigation error you're going to have. An additional trend for each GPS update rate is that having an additional low frequency aiding sensor for velocity improves the state estimation performance.

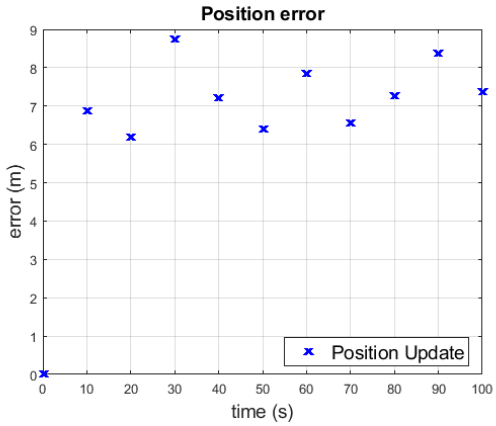


Figure 13: Position error at 0.1 Hz GPS update w/o velocity aiding.

Figure 13 shows an example of the error plotted every time there is a GPS update. Not that the error that is calculated is based on the previous epoch state estimate as soon as there is a new GPS measurement available. Figure 14 shows the same error plot but with velocity aiding.

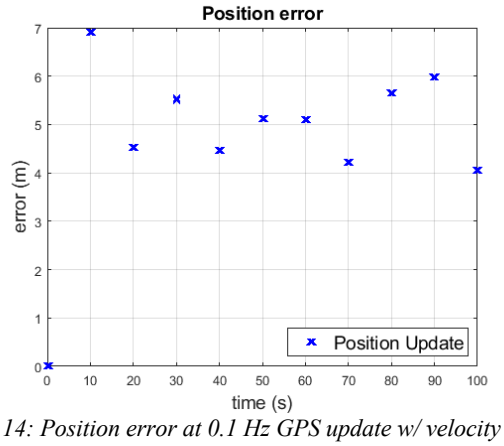


Figure 14: Position error at 0.1 Hz GPS update w/ velocity aiding.

The trend is clear looking at Figure 13 and 14 that an external velocity input will increase the accuracy of your dead reckoning solution. As the GPS updates become very infrequent the dead reckoning solution does not perform that well over all without velocity aiding. Looking at Table 1 again we see that for our slowest GPS update rate of 0.005 Hz the algorithm has a large error per percent distance travelled. However, for that same update rate but with velocity data available the error per percent distance travelled his cut in half. Additional data sets for other GPS update rates are available in Appendix VIII.

Additionally, it was noticed that for most simulation runs the error between the initial starting point and the first low frequency aiding measurement was the highest. We believe this is due to the way the Kalman covariances and gains are initialized until being recalculated at a low frequency position update. This is a point to note when implementing into a real time system and making sure to have optimal initial covariances and gains.

V. Conclusion

This study has successfully demonstrated the viability of an IMU-based dead reckoning system as an alternative to GNSS for autonomous underwater vehicle (AUV) localization, particularly in environments where GNSS is unavailable. Through the implementation of a dead reckoning algorithm utilizing both high-precision and low-precision IMUs, along with simulated DVL updates, it has been shown that robust position estimation is achievable.

Key findings from the experiments include the observation that the performance difference between high and low precision IMUs was negligible, indicating that for the purposes of dead reckoning in AUVs, a lower precision—and potentially less costly—sensor may suffice. This supports previous research suggesting that when fused with other sensor data, the precision of the IMU has less impact on the overall accuracy of the system. The study further explored the impact of varying DVL update frequencies on the accuracy of the dead reckoning algorithm. An intriguing outcome of this research was the revelation that the absence of DVL updates yielded better position estimations than scenarios with infrequent updates. This counterintuitive result suggests that sporadic velocity updates may introduce a degree of uncertainty, potentially disrupting the consistency of the dead reckoning process. It's a phenomenon that warrants further exploration through simulation as well as in real world testing environments. On the other hand, the results demonstrated a marked improvement in position accuracy with increased frequency of DVL updates, affirming the value of more regular feedback in refining the dead reckoning algorithm's output. Another significant outcome of this project is the relationship between GNSS update frequency and localization accuracy. Consistent with expectations, an increase in GNSS update frequency correlated with a decrease in RMSE.

The feedback Kalman filter dead reckoning algorithm was found to successfully minimize error and make system corrections upon low frequency position updates. This type of algorithm is important to making robust INS systems that are able to provide good position estimates in the event of periodic loss of position updates. It was found that implementing an external velocity source into the filter did reduce the overall error of position estimates. This is important particularly for AUV systems that need to traverse long duration with very low frequency position updates but have access to accurate velocity measurements from a DVL.

The contributions of this study extend the existing body of knowledge on autonomous vehicle localization by providing practical insights into the sensor fusion of IMUs and DVLs. Moreover, the findings offer guidance for the design of AUV navigation systems, suggesting avenues for cost savings without compromising on the performance of the localization system. As AUVs continue to be deployed for increasingly complex missions, the demand for effective and reliable localization systems that can operate independently of GNSS will grow. The outcomes of this research serve as a stepping stone toward the development of more advanced navigation systems that leverage the strengths of dead reckoning and sensor fusion techniques.

References

[1] Arsan, Taner. (2020). Accurate indoor positioning with ultra-wide band sensors. Turkish Journal of Electrical Engineering and Computer Sciences. 2020. 1014. 10.3906/elk-1911-79.

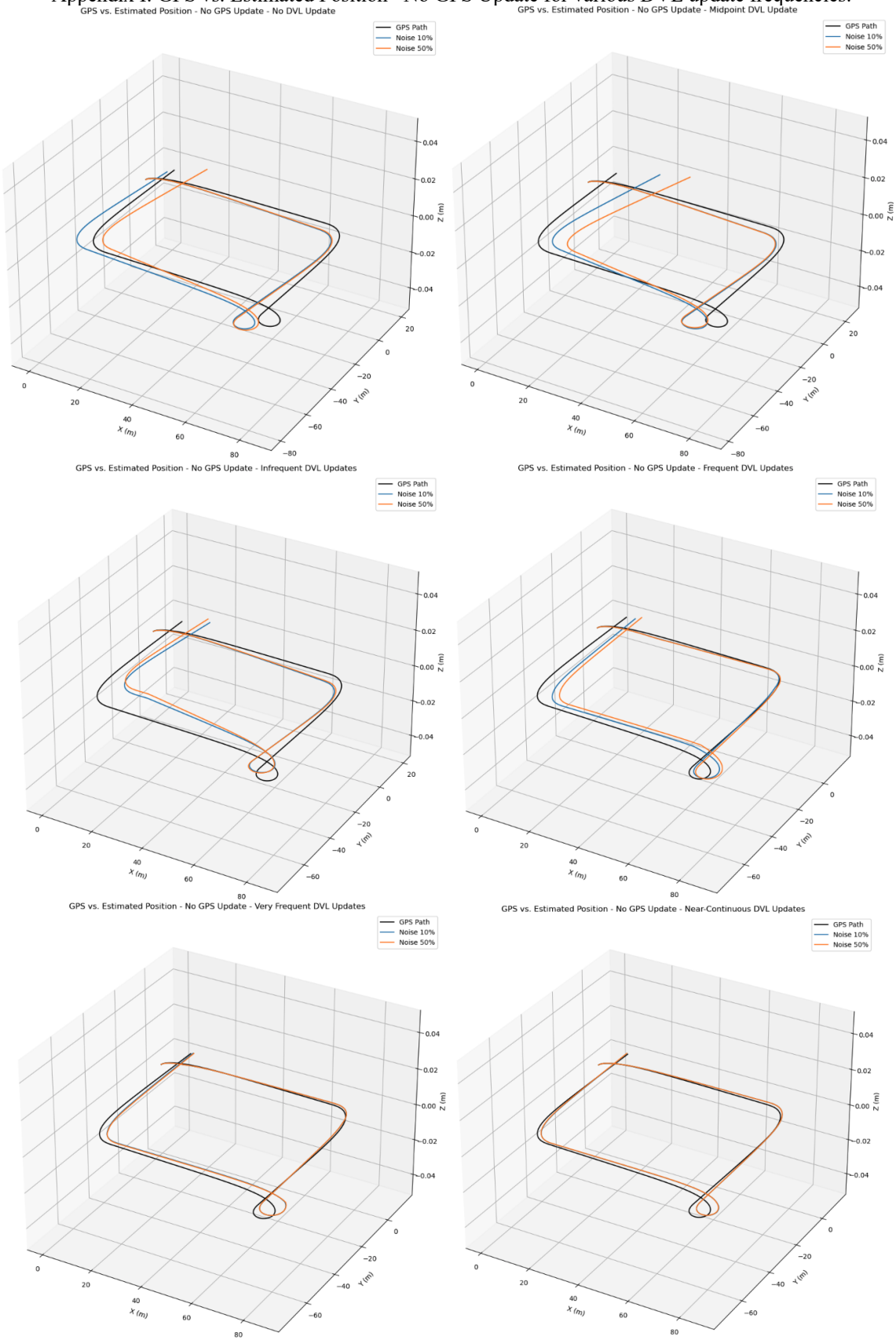
[2] I. Toy, A. Durdu and A. Yusefi, "Improved Dead Reckoning Localization using IMU Sensor," 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2022, pp. 1-5, doi: 10.1109/ISETC56213.2022.10010239.

[3] Kalman, R. E. (March 1, 1960). "A New Approach to Linear Filtering and Prediction Problems." ASME. J. Basic Eng. March 1960; 82(1): 35–45. https://doi.org/10.1115/1.3662552

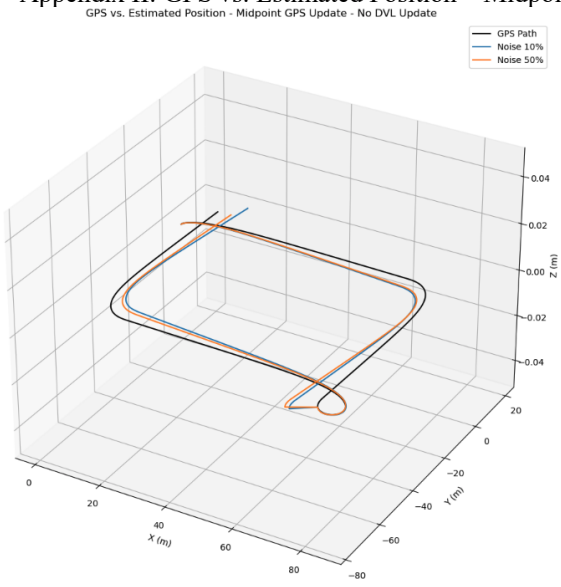
[4] Y. Geng, R. Martins and J. Sousa, "Accuracy analysis of DVL/IMU/magnetometer integrated navigation system using different IMUs in AUV," IEEE ICCA 2010, Xiamen, China, 2010, pp. 516-521, doi: 10.1109/ICCA.2010.5524143.

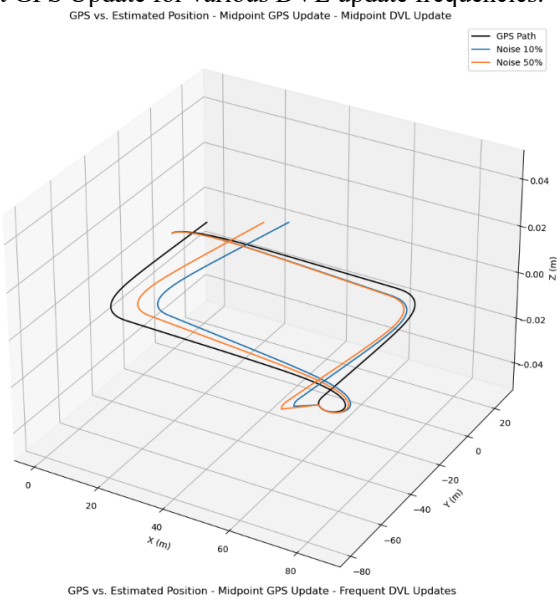
[5] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control,* 2nd ed. Hoboken, NJ, USA: Wiley, 2021.

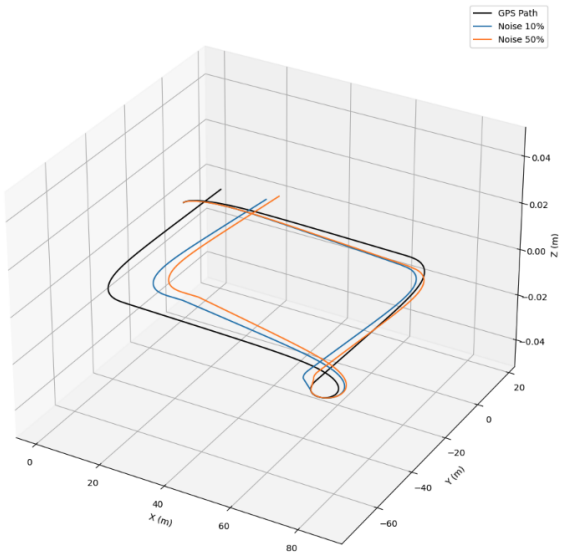
Appendix I: GPS vs. Estimated Position –No GPS Update for various DVL update frequencies.

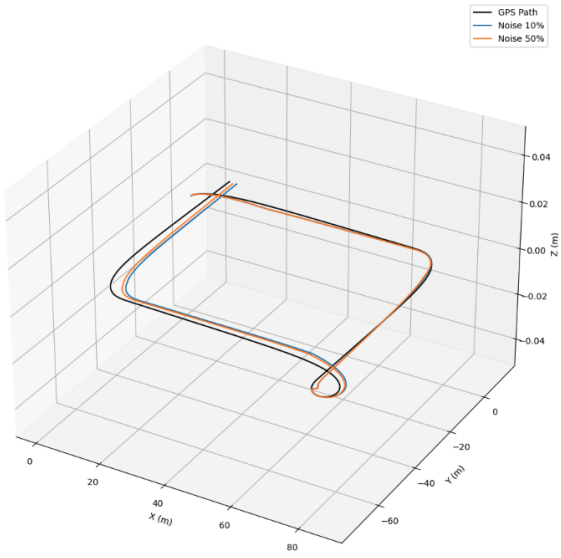


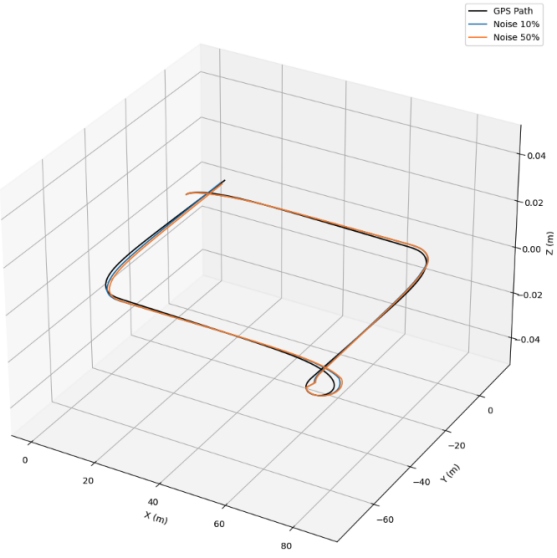
Appendix II: GPS vs. Estimated Position – Midpoint GPS Update for various DVL update frequencies.

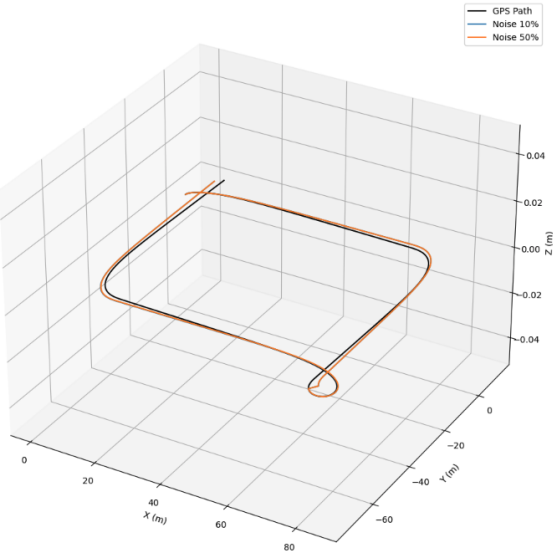




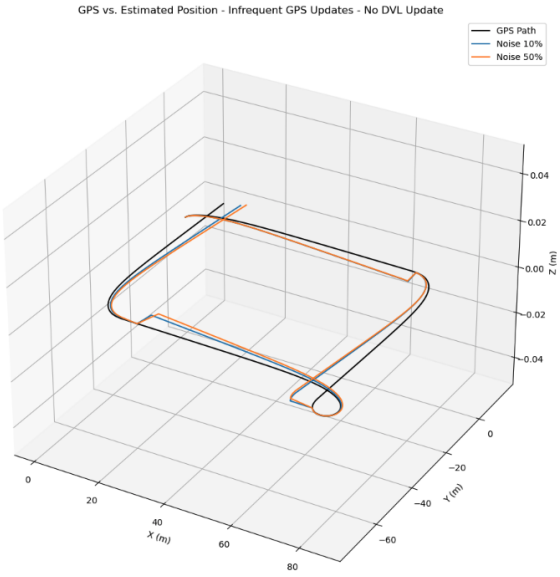


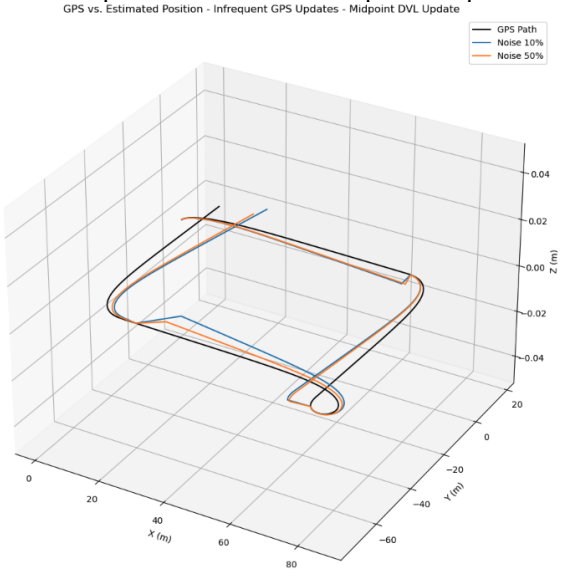


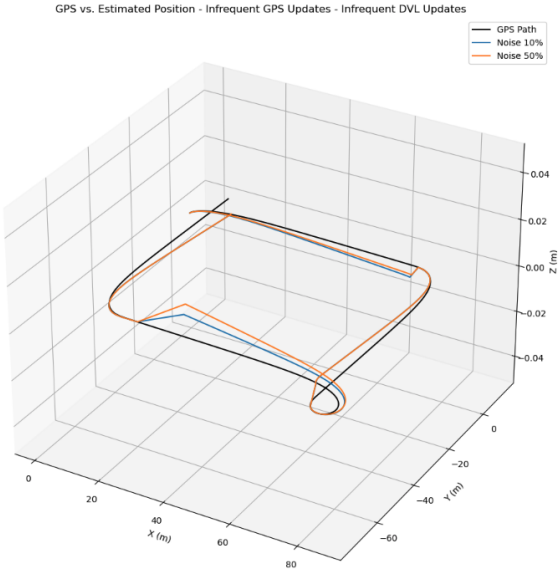


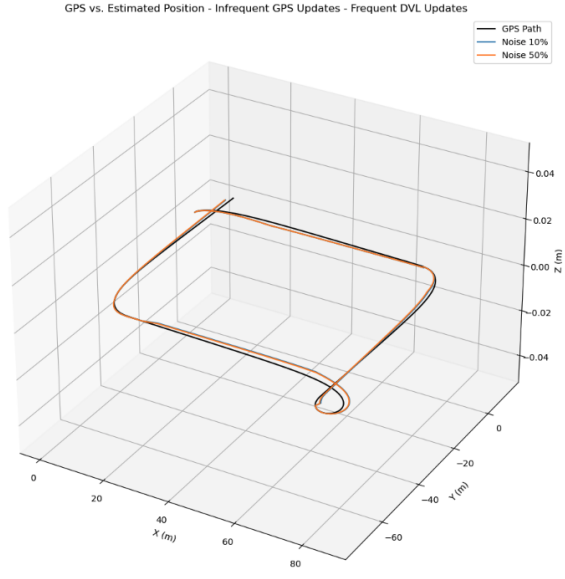


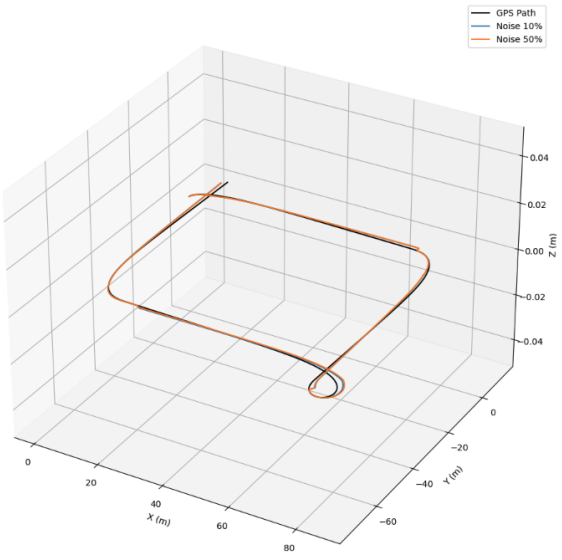
Appendix III. GPS vs. Estimated Position – Infrequent GPS Update for various DVL update frequencies.

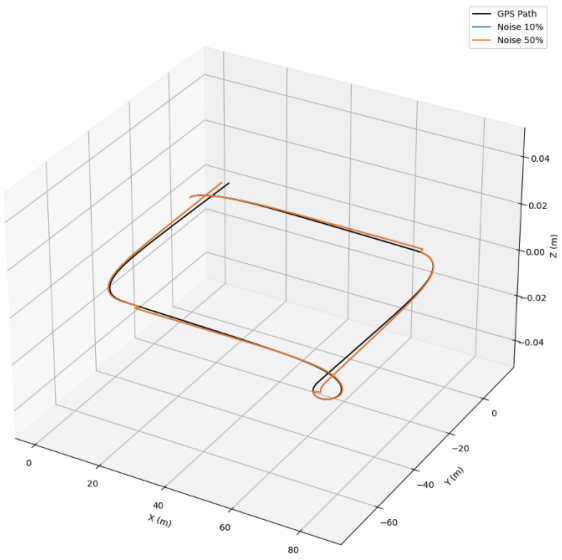




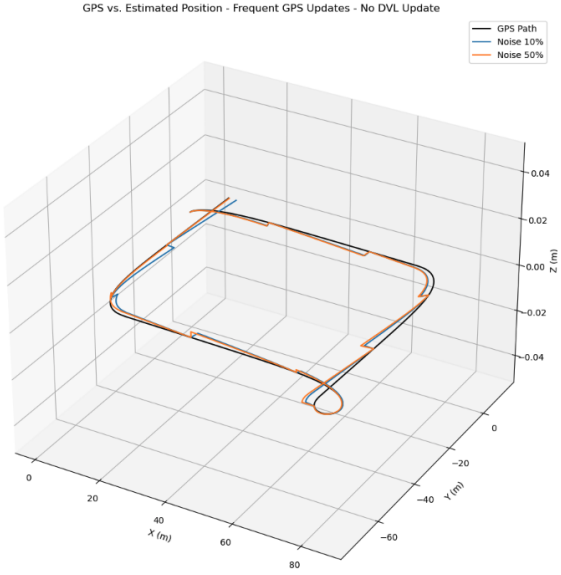


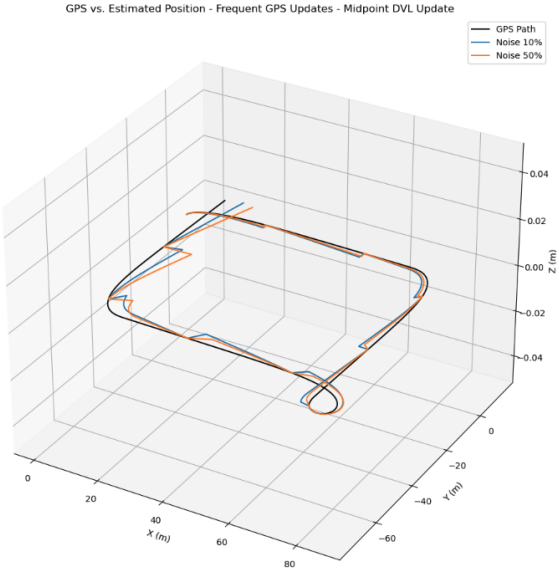


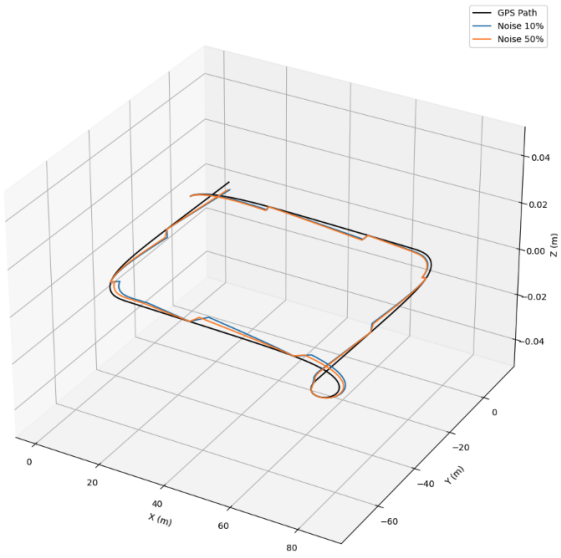


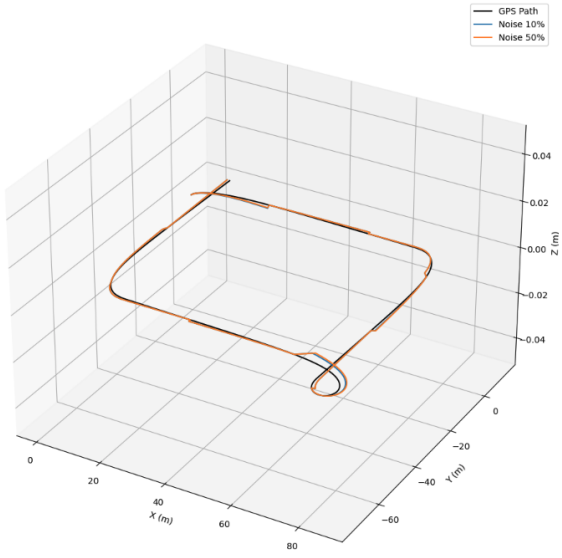


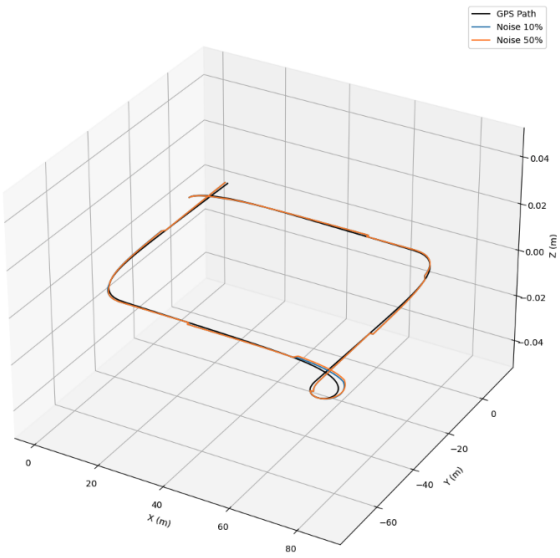
Appendix IV. GPS vs. Estimated Position – Frequent GPS Update for various DVL update frequencies.

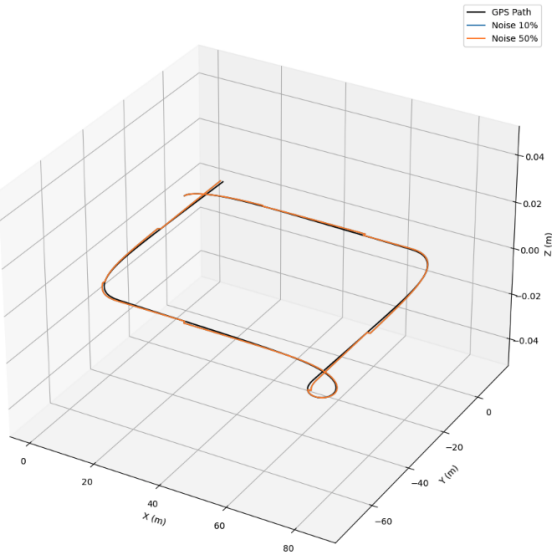




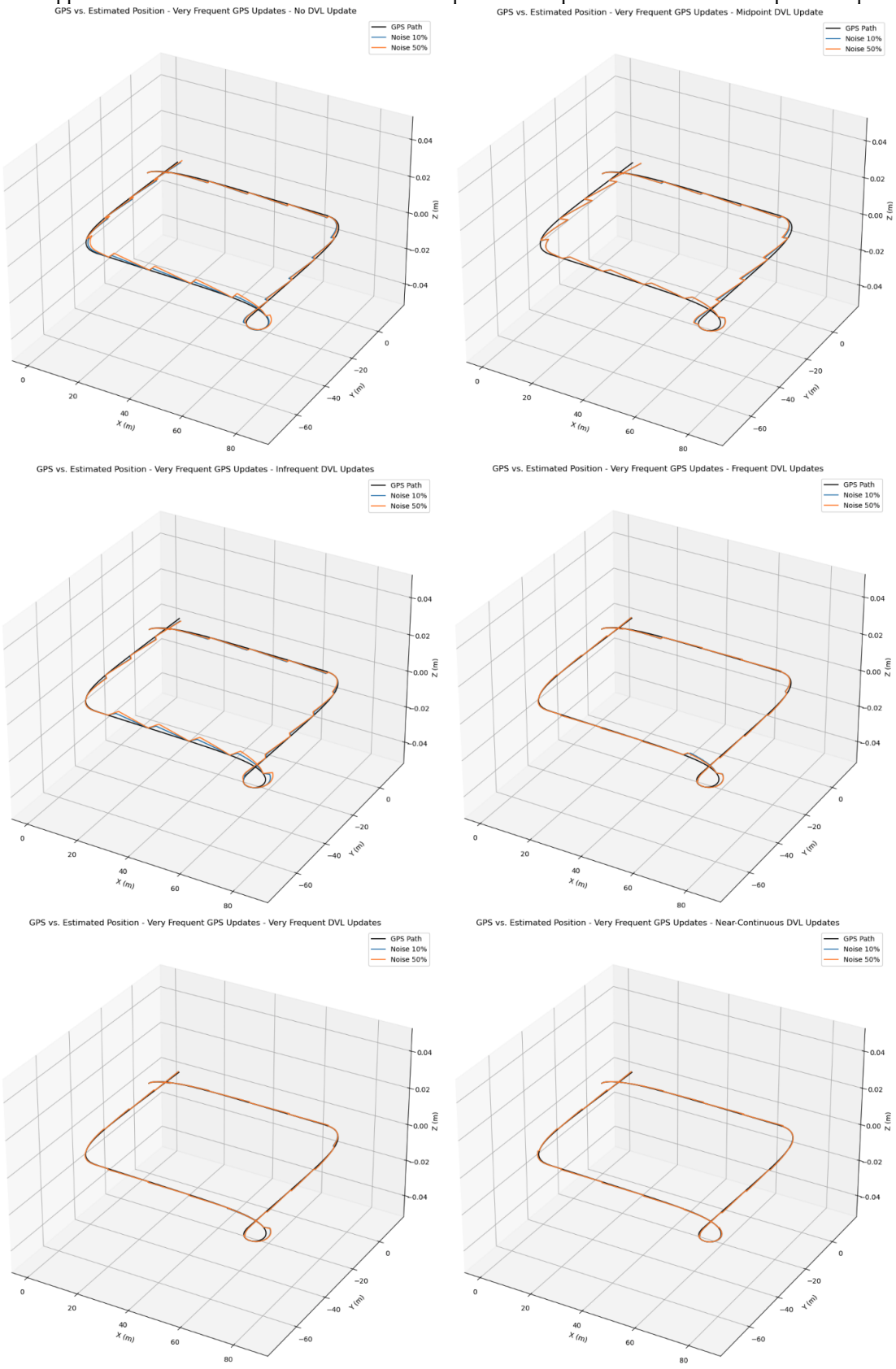




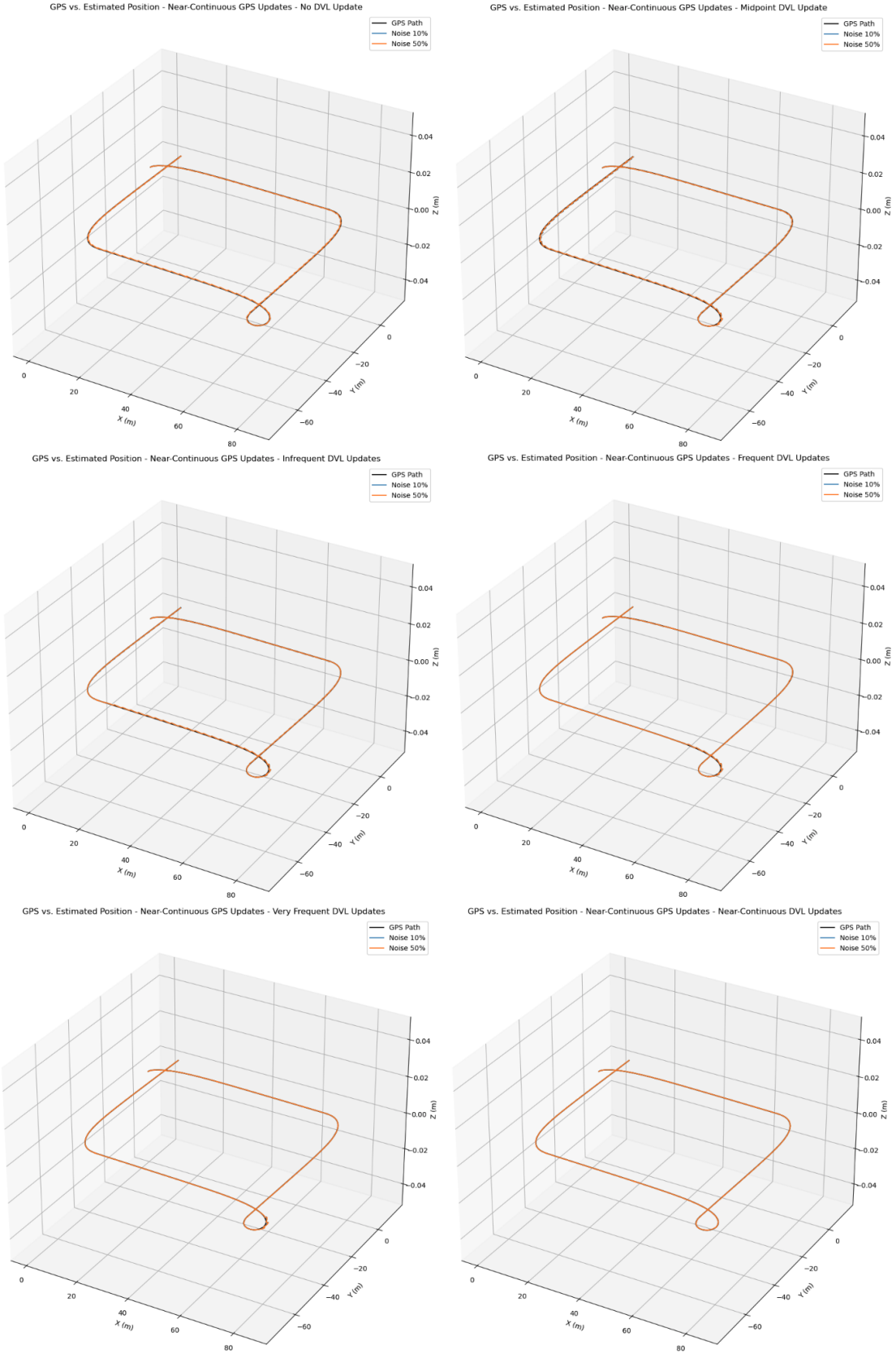




Appendix V. GPS vs. Estimated Position – Frequent GPS Update for various DVL update frequencies.



Appendix VI. GPS vs. Estimated Position – Frequent GPS Update for various DVL update frequencies.



Appendix VII. GPS vs. Estimated Position – Error-state Kalman Filter for differing GPS update frequencies

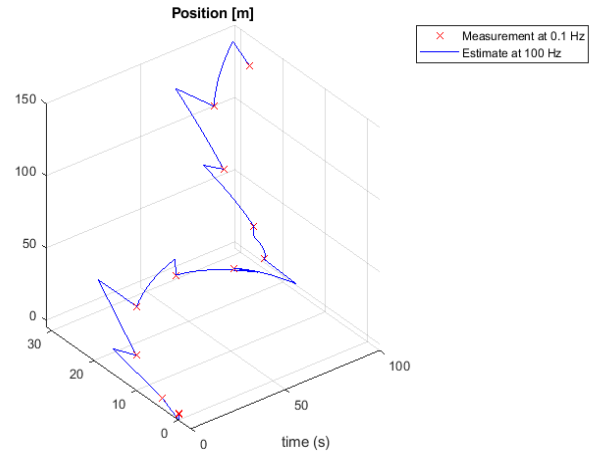


Figure 1: 0.1 Hz GPS update frequency w/o velocity data.

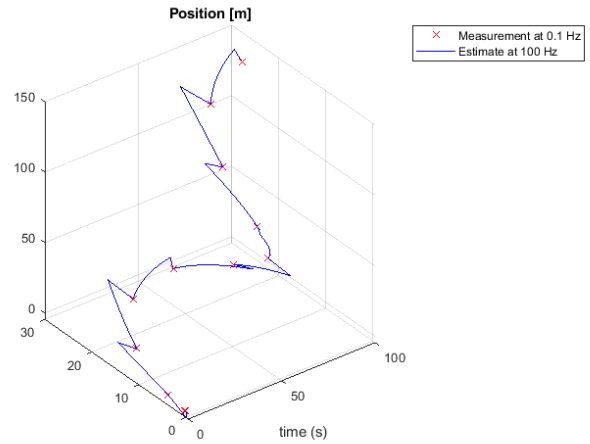


Figure 2: 0.1 Hz GPS update frequency w/ velocity data.

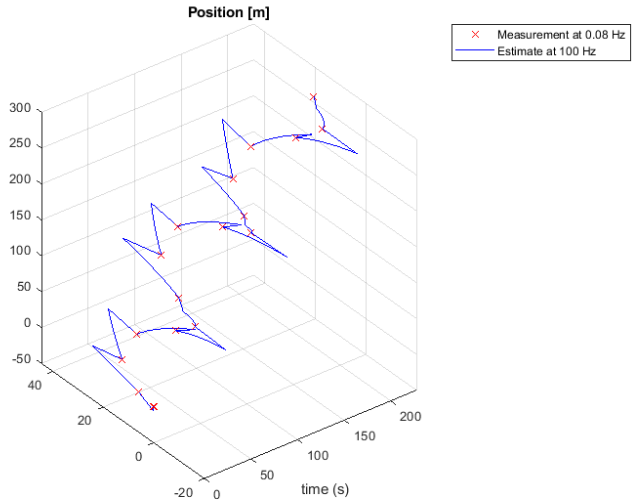


Figure 3: 0.08Hz GPS update frequency w/o velocity data.

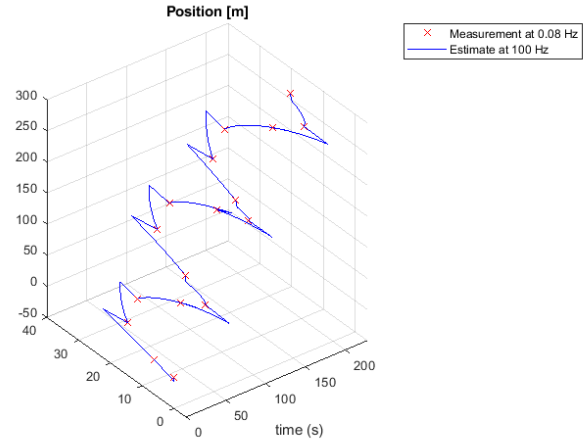


Figure 4: 0.08 Hz GPS update frequency w/ velocity data.

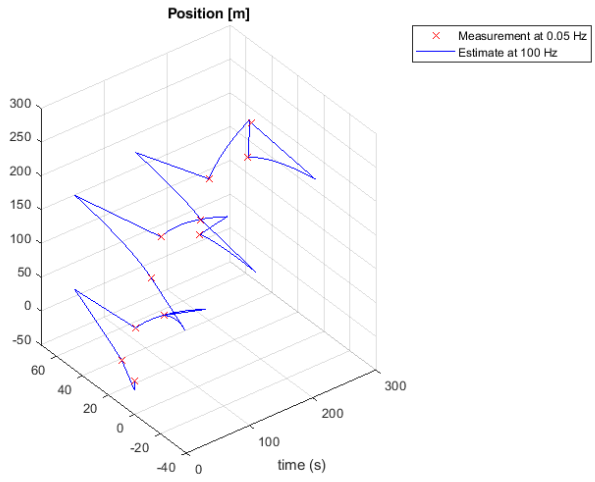


Figure 5: 0.05 Hz GPS update frequency w/o velocity data.

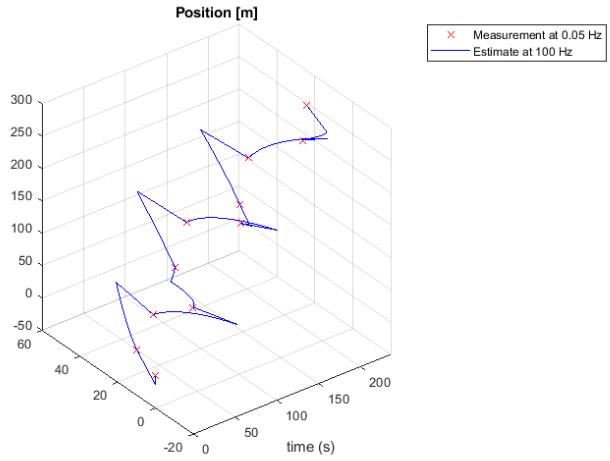


Figure 6: 0.05 Hz GPS update frequency w/o velocity data.

Appendix VIII. Error – Error-state Kalman Filter for differing GPS update frequencies (w/o velocity aiding)

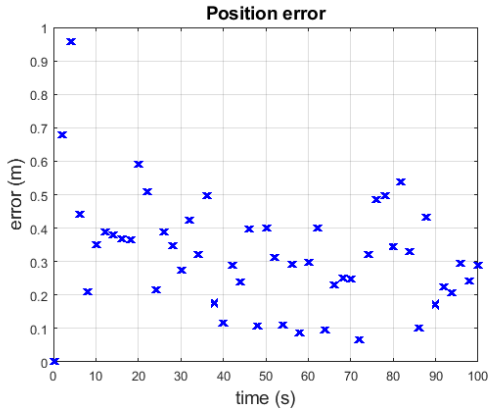


Figure 1: Position error at 0.5 Hz GPS update frequency w/o velocity aiding

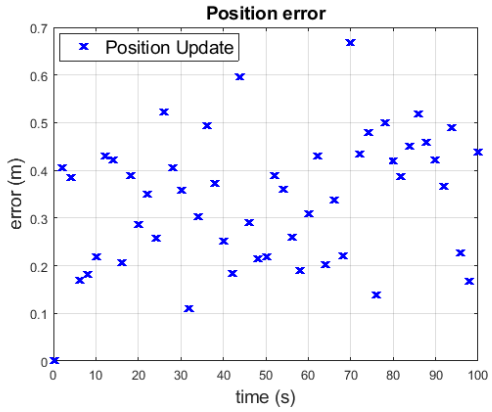


Figure 2: Position error at 0.5 Hz GPS update frequency w/ velocity aiding

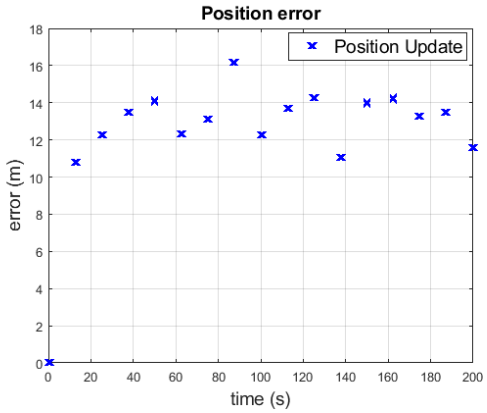


Figure 3: Position error at 0.08 Hz GPS update frequency w/o velocity aiding

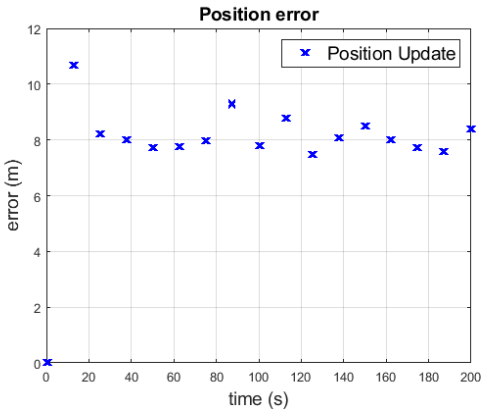


Figure 4: Position error at 0.08 Hz GPS update frequency w/ velocity aiding

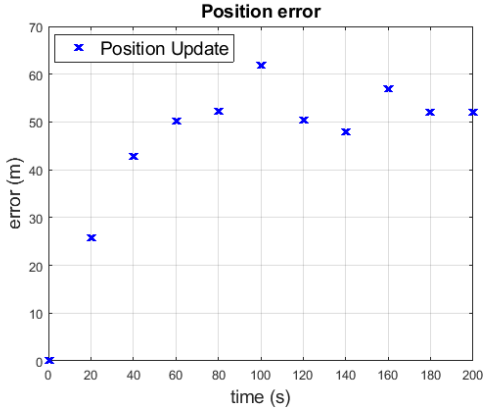


Figure 5: Position error at 0.05 Hz GPS update frequency w/o velocity aiding

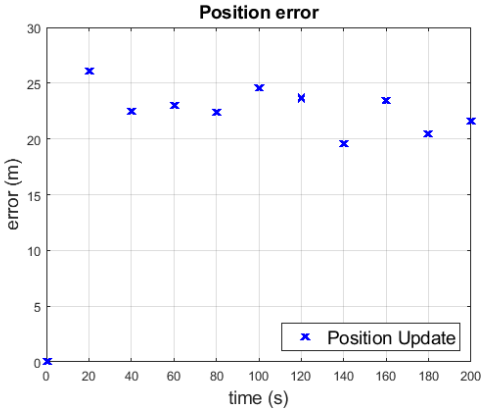


Figure 6: Position error at 0.05 Hz GPS update frequency w/ velocity aiding

Appendix VII. Error – Team Member Contributions

Ryan Gochar -

- Final Report
  - Abstract
  - Introduction
  - Methodology: Dataset, Open Loop DR Algorithm
  - Results
  - Analysis
  - Conclusion
  - Appendices 1 – 6
- Code
  - All python implementation of:
  - DR Algorithm
  - Kalman Filter
  - GPS Updates
  - DVL Updates
  - All plots generated in Python.
  - Error Analysis (python + excel)
- Presentations:
  - Dead reckoning
  - Kalman Filter
  - Methodology: Open loop DR
  - Results: Open Loop DR
- Git
  - Created repository.
  - Created Readme
  - Pushed all notebooks.

Brandon Davis –

- Final Report
  - Introduction
  - Methodology: Error state Kalman filter
  - Results: Error state Kalman filter
  - Analysis: Error state Kalman filter
  - Conclusion
  - Appendices 7-8
- Code
  - Python DR
  - Python sensor simulation
  - Python sensor data transforms
  - MATLAB error state Kalman filter
- Presentation:
  - Dead reckoning
  - Error state Kalman filter
  - Background
  - Methodology
  - Results
- Git
  - Contribute to README
  - Contribute to code.