AO35: Trajectory Reconstruction Using Inertial Measurment Units And The Implications For Aerobots On Venus

May 24, 2021

Abstract

The trajectory tracking requirements of balloon based aerobot missions on Venus are discussed, along with the role Inertial Measurement Units and Deadreckoning can play in reconstructing these trajectories. Methods of trajectory reconstruction are then outlined and inplimented in a Python work enviroment. A 6 axes LSM6SD3 is then calibrated and used to test these reconstruction methods, with an 8 fold performance benefit being achieved with the use of an Extended Kalman Filter over simple reconstruction methods. The implications of these finding for aerobots missions on Venus is then discussed.

1 Introduction

In 1985 the VeGa 1 and VeGa 2 missions each deployed a balloon based Aerobot into the upper cloud layer of Venus [1]. Since these missions, there has been a long line of proposals to again deploy an Aerobot into Venus's atmosphere. These proposals include: the 2020 Venus Flagship Proposal (2020 Proposal) [2], the 2010 European Venus Explorer (EVE) mission proposal [3], and New-Frontiers Class In-Situ Exploration of Venus: The Venus Climate and Geophysics Mission Concept [4]. These missions aim to deploy an aerobot carying a suite of scientific instuments designed to collect geophysical, meteorological, and cloud composition data form the turbulent upper cloud layer of Venus.

A key requirment of these missions is the ability to track the location of the Aerobot within the atmosphere. Tracking the trajectory of the aerobot allows for both the calculation of wind speed and direction [5], and the ability to tie the collected geophysical and compositional data to a specific location, thus increasing the scientific potential of these measurments. The Vega missions were tracked using an array of 20 Earth based radio telescopes with the 2020 and EVE proposals also set to rely on this method. However, the coverage provided by Earth based telescopes is limited to when the aerobot is on the near side of Venus. The 2020 proposal included 3 orbiting satellites to provide additional coverage when the aerobot is obscured from earth based observations. However, even with 3 satellites, permanent coverage cannot be provided and there will be periods of up to 1 day when the Aerobot goes untracked.

The 2020 proposal attempts to address this issue by including an Inertial Measurement Unit (IMU) in the aerobot cradle. The acceleration and angular rate data collected by the IMU would be used to reconstruct the trajectory of the aerobot in a process known as dead reckoning [6] [7]. Additionally, the accelerometer data can be used to provide a finer resolution analysis of the turbulence within the atmosphere than eternal tracking alone.

In order to design a system capable of achieving the desired accuracy for the aerobot, an understanding of how different factors affect the accuracy and stability of the reconstructions is required. This report will examine the applicable trajectory reconstruction methods, the sources and growth of errors, and the possible mitigation steps that can be taken, such as data fusion and the use of an Extended Kalman Filter (EKF). The implications of the findings for trajectory reconstruction of Aerobots on Venus will then be discussed alongside areas in which further research could be conducted to aid in the construction and testing of the aerobots system.

To aid the investigation of the trajectory reconstruction process, a physical investigation was undertaken using a 6 axis LSM6SD3 IMU [8] embedded on an Arduino 33Iot microcontroller board. First, the characteristics of the sensors noise were examined, before the optimisation procedure outlined in [9]. was used to calibrate the accelerometer and gyroscope triads. An array of trajectory reconstruction algorithms, each relying on a different approach, were then constructed, and their relative performances analysed.

2 Method

2.1 Physical Apparatus

The IMU used for the investigation was the 6 axis LSM6DS3 [8], which contains two Micro Electro-Mechanical Sensor (MEMS) triads, one to measure acceleration and the other angular velocity. This sensor was chosen as it can be purchased mounted to an Arduino 33Iot microcontroller board that can poll the sensor and export the data in real time to an external computer. The LSM6DS3 was driven using the $< Arduino\_LSM6DS3 >$ library [10] which set the sampling frequency to 100HZ, the accelerometer triads range to $\pm$4g with a resolution of $\pm$0.122mg, and the gyroscope triads range to $\pm$2000dps with a resolution of $\pm$70mdps.

The 33iot was placed fixed to a breadboard alongside a 3.7V LI-ION battery used to provide 3V power to the microcontroller. The 33iot could be configured to either export data in real time via a Bluetooth connection, Wi-Fi connection, or by using a USB cable.

2.2 Testing

The tests performed fell into two broad categories: stationary tests and moving tests, where the sensor was put through a series of translations and rotations. The 33iot was configured to continuously poll the LSM6SD3 at a frequency of 100Hz and export the data over a communications link to an external computer. The external computer ran a script in Python to initialise the communications link, collect the polled data, and write it to a CSV file.

Test type	Properties	Purpouse
Stationary	Sensor left at rest on a stable surface.	Calibrating sensor measurement model parameters and measuring noise distributions.
Motion Testing	The sensor is moved through a known and well defined sequence of rotations and/or translations.	Tuning algorithms to accurately reconstruct known trajectories.

Table 1: Descriptions of the properties and uses of each of the testing styles.

3 Modelling, Algorithms and Code

3.1 Modelling the Problem

Should the initial attitude and position of an IMU be determined, the angular rates measured by the sensor can be integrated to estimate the attitude of the sensor at any proceeding time. In turn, the measured accelerations can be integrated twice to find the position of the sensor, and so its trajectory can be tracked. The trajectory reconstruction problem requires three reference frames:

The Body Frame: Defined as having the accelerometer triad at its origin with the axes aligned to those of the sensor.

The Inertial Frame: Defined as stationary within space. The sensor measures the accelerations and angular rates relative to this frame. In the Venus case, it would have its origin at the centre of the planet and its axes fixed relative to the stars.

The Navigation Frame: This describes the space within which the position of the sensor needs to be determined. In the Venus case, this would be the geocentric common ellipsoid for Venus that rotates with the planet and has the unit gravity vector parallel to its surface.

The maximum length scale of the completed tests was under 10m and thus the navigation frame was approximated as euclidean during this investigation. Additionally, the rotation of the earth was neglected and the navigation frame was taken fixed with respect to the Inertial frame. This was possible as the angular velocity of the earth, 4×10^{-3} degrees/s, falls below the resolution of the gyroscope, and the expected magnitudes of the coriolis and centripetal acceleration, $10^{-5} m/s^2$ (with a typical maximum velocity of 10 m/s) and $10^{-4} m/s^2$ respectively, fall below the resolution of the accelerometer. These approximations, however, will not be applicable to an aerobot on Venus as they will be expected to circumnavigate the planet multiple times.

With the above approximations considered, the navigation frame was defined as having the Earth's gravity vector perpendicular to the XY plane at the origin - the point that the trajectory reconstruction was initialised. The X and Y axis of the navigation frame were then aligned with the projection of the X and Y axis of the body frame on the XY plane of the navigation frame. Due to the Euclidean and non-rotating approximations, the specific forces could be directly rotated into the navigation frame before integration with no further transformations required.

3.1.1 State Vectors

During this investigation the attitude of the sensor was described using Unit Quaternions as state vectors. Equation 1 shows how the quaternion state of the sensor Q_{nb} can be used to rotate the quantities measured in the body frame, Q_{body}, into the navigation frame, Q_{nav}.

$$Q_{nav} = Q_{nb} * Q_{body} * Q_{nb}^{T} \tag{1}$$

Here, $*$ is the quaternion multiplication operation, and Q_{nb}^{T} the quaternion conjugate of Q_{nb}. The quaternions Q_{body} and Q_{nav} take the form $(0, iV_{body})$ and $(0, iV_{nav})$ respectively, where i is the imaginary unit. For data visualisation purposes, the attitude has been represented using Euler angles with the (XYZ) convention where the roll, pitch, and yaw (L, M, and N) angles represent rotations about the X, Y, and Z axes respectively. The code does not rely upon Euler angles for functionality so the convention used can be adjusted by altering the function eul_from_q in the $\_Functions.py$ script.

3.2 Reconstruction Methods

3.2.1 Defining the Initial Sate

The initial position and velocity of the sensor can be trivially defined as zero for a sensor starting at rest. However, defining the initial attitude of the sensor is a non-trivial task with implications for the stability of the trajectory reconstruction. This is because roll and pitch errors result in a residual of the gravitational acceleration remaining in the X and Y components of the linear acceleration in the navigation frame. This is a highly sensitive problem, where, for the Venus case (with a surface gravity of $8.858m/s^2$), an accuracy of 10^{-3} degrees is required to know the position to an accuracy of 1km after an hour.

The initial roll and pitch were determined using an at rest calibration period at the beginning of each data set. At rest, the measured accelerations are purely gravitational in origin, and thus the rotation between the body frame and the navigation frame can be calculated as the rotation between the acceleration measured by the sensor $\vec{a}$ and the known gravity vector in the navigation frame $\vec{g}$. This rotation can be calculated using Equation 2. The accuracy of the initial attitude estimate can be improved by calculating an attitude quaternion, using Equation 2, for each data point during the calibration phase, and then the average of these attitude quaternions was taken as the final estimate. The averaging of the quaternions was performed using the method in Appendix A.1 [11].

$$Q_{nb} = \left(cos\left(\frac{\phi}{2}\right), i\vec{n}sin\left(\frac{\phi}{2}\right)\right) \tag{2}$$

where:

$$\vec{n} = \frac{\vec{a}\times\vec{g}}{||\vec{a}\times\vec{g}||} \tag{3}$$ and $$\phi = arctan\left(\frac{||\vec{a}\times\vec{g}||}{\vec{a}\cdot\vec{g}}\right) \tag{4}$$

3.2.2 Integrating Gyroscope Measurements to find Attitude

The attitude estimate of the sensor was updated for each time step by numerically intergrating the angular velocity measurements. For quaternion states, a correct choice of intergration algorithm would be the normalised fourth order Runga-Kutta method [12]. The updated attitude estimate Q_{k+1} is caculated from the previous estimate Q_k using Equation 5.

$$Q_{k+1} = Q_k + \frac{\delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{5}$$

where:

$$k_i = f(q_i, t + c_i\delta t) \tag{6}$$ $$q_i = Q_k + \delta t\sum_{j=1}^{i-1} a_{ij}k_j \tag{7}$$ $$f(q_i, t) = \frac{1}{2}\Omega(X_{g,t})q_i \tag{8}$$

In these equations, $X_{g,t}$ is the angular rate at time t and δt is the time step, the rate at time $(t + c_i\delta t)$ is found by linearly interpolating between the measurments at time t and $t + \delta t$. The definition of $\Omega(X_{g,t})$ can be found in Apendix A.2. This method of intergration does not preserve the unitary nature of rotation quaternions and thus the attitude estimate Q_{k+1} must be renormalised using Equation 9 before it can be used.

$$Q_{k+1}^{norm} = \frac{Q_{k+1}}{||Q_{k+1}||} \tag{9}$$

3.2.3 Integrating Accelerometer Measurements for Velocity and Position

Once the attitude of the sensor was calculated, the acceleration measured in the body frame, $\vec{a}_{body}$, was rotated into the navigation frame using Equation 1. The gravitational contribution of $\vec{g}m/s^2$ was then removed from $|vecA)nav$ to find the linear accelerations in the navigation frame. These accelerations can then be integrated for each time step, once to find the change of velocity in the navigation frame and a second time to find the displacement of the sensor.

3.2.4 Modelling Sensor Measurements

The accuracy of the above algorithms relies on the measurements reported by the sensor being the true values. However, due to fabrication and noise limitations, the measurements taken by the sensor need to be adjusted before they can be used. The fabrication limitations of a MEMS clusters mean that each individual triad will be non-orthogonal, have zero offset errors, and have scaling coefficients unique to it. Additionally, the accelerometer and gyroscope triads will not be perfectly aligned.

These errors on the IMU readings were corrected for using the model in Equation 10 [9] where the index i can represent either the accelerometer triad (a) or the gyroscope triad (g). Here, $Y_{i,t}$ is the measurement vector at time t and $X_{i,t}$ is the true vector in the body frame.

$$X_{i,t} + e_{i,t} = (T_i K_i [Y_{i,t} - \delta_i]) \tag{10}$$

δ_i acounts for the zero offset error; K_i individually scales the readings for each measurment axes; and T_i transforms the readings into an orthogonal frame and, in the gyroscope case, rotates the frame to match the acelerometer's frame. An additional random error $e_{i,t}$ is included in the measurement model to represent the random noise present in the readings. As shown in Equation 11, this noise was modelled as Gaussian with a variance of σ_i and a time varying Zero offset $c_{i,t}$.

$$e_{i,t} = N(c_{i,t}, \sigma_i) \tag{11}$$

3.2.5 Extended Kalman Filtering (EKF)

The random errors $e_{i,t}$ on the sensor readings cause the uncertainty on the calculated position and attitude to grow with time. In particular, the drift of the gyroscope based attitude estimate with time is a major source of error for the position estimates due to a residual gravitational acceleration remaining in the calculated linear accelerations.

Extended Kalman filtering is a data fusion technique that allows for real time attitude estimates derived from sources other than integration of the Gyroscope readings, to be used to supress the drift in the attitude estimates caused by the random errors. The Filter weights the attitude estimates together, with the weights dependant on the accuracy of each individual estimate, and then from this weighted sum derives a mean attitude prediction.

In this investigation, an EKF was used to weight the gyroscope attitude prediction against one obtained from the accelerometer data. The accelerometer data was analysed for periods at which the magnitude of the linear acceleration of the sensor was calculated to be much smaller than the magnitude of the earth's gravity. Therfore, the attitude of the sensor could be calculated using the method described in Equation 2. This accelerometer based attitude estimate was then weighted against the one calculated by intergration of the gyroscope mesurements to form a new more accurate estimate.

The attitudes were averaged by taking the outer product of each of the attitude estimate quaternions with themselves and then finding the weighted sum of each of these outer products, as shown in Equation 12, to form the 4x4 matrix Q_{mat}. The quaternion that represents this matrix can then be found as the Eigen vector, corresponding to the largest Eigen value of Q_{mat}, as detailed in Appendix A.1.

$$Q_{mat} = [q_{gyro} * q_{gyro}^T]\beta_{gyro} + [q_{grav} * q_{grav}^T]\beta_{grav} + \sum_i [q_i * q_i^T]\beta_i \tag{12}$$

3.2.6 Algorithms For Iterative Calculation of Position and Attitude

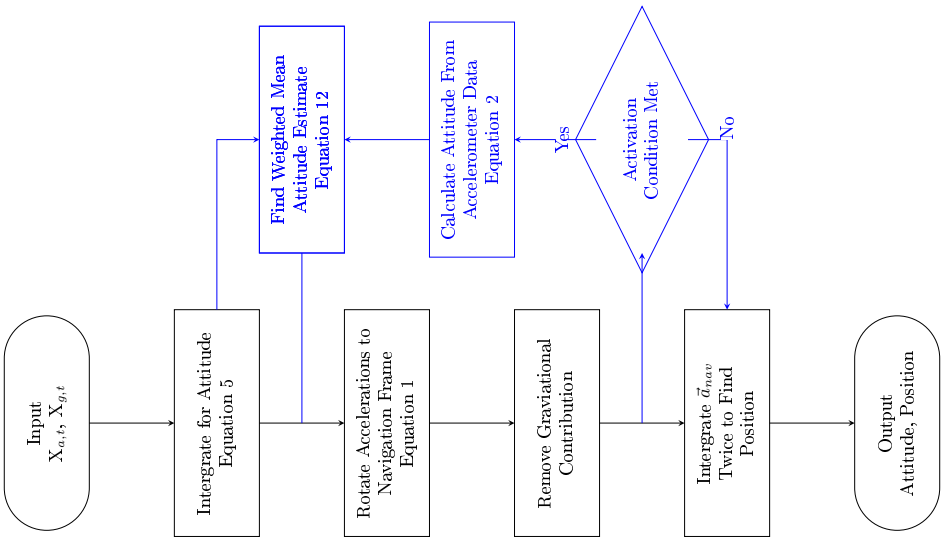

Figure 1: A flow chart displaying the algorithm to iteratively calculate the attitude and position of the centre. The blue path is only followed by EKF on the first pass, whilst the simple algorithm exclusivley follows the black path.

3.3 Code and Data Processing

The data analysis, sensor characterisation, and trajectory reconstructions were carried out using a range of scripts written in Python, utilising the methods described in Section 3.2. The code files can be found at [13] and descriptions outlining the functionality of each script can be found in Appendix B.

4 Sensor Characterisation and Calibration

The first phase of testing and analysis focused on finding the calibration coefficients for the measurement model in Section 3.2.4. The code in *_calibration.ipynb* utilised the data set *calibration_data.csv* and the minimisation procedure outlined in "A Robust and Easy to Implement Method for IMU Calibration without External Equipments" [9] to determine the calibration coefficients.

4.1 Noise Characterisation

Initially, an analysis of the sensor noise was completed in order to understand how the errors introduced by the noise would affect the accuracy of the found calibration paramaters. A data set was collected whilst the sensor was left stationary for approximately 5.5 hours, during which 2 million readings were collected at a frequency of 100HZ. The Allan variances of both the gyroscope and the accelerometer triads were calculated as described in [14] and subsequently plotted in Figure 2. The Allan variance of both the gyroscope and the accelerometer reach a minimum at 60 seconds, however past this point the gyroscopes allan variance begins to climb indicating a time dependant zero offset error in the readings. Consequently, to minimise the effect of these random errors on the estimated initial attitude of the sensor or on the calculated zero offset errors of the gyroscope, the length of the stationary calibration phase was set to be 50 seconds. The increase in the Allan variance of the gyroscope past 60 seconds indicates a time varying zero offset error, thus the offset must be individually calculated at the beggining of each reconstruction and to minimise drift errors in the attitude estimates the reconstrcution should be kept under 60 seconds. Finally the distribution of the random noise error was analysed by using a 60 second (6000 reading) segment of the data set. A Gaussian profile was individually fitted to each axis of the accelerometer's and gyroscope's triads. The variances calculated for each triad can be found in Table 2.

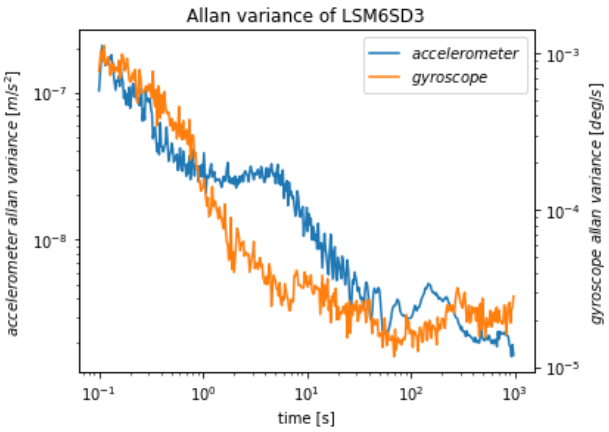

Figure 2: loglog plot of the allan variance of the accelerometer and gyroscope triads.

4.2 Accelerometer Calibration

The accelerometer coefficients were determined by first identifying a number of stationary periods within the data set, then averaging the acceleration measurements for each of these periods and producing a set of average acceleration vectors $Y^s_{a,i}$ from the stationary periods. The cost function in Equation 13 was then minimised using the trust constrained minimisation procedure in the SciPy library [15]. Here, the vector set $X^s_{a,i}$ is the vector set $Y^s_{a,i}$, adjusted using the measurement model in Section 3.2.4, and $|g|$ is the magnitude of the local gravity vector at the latitude and longitude of calibration. The optimal values for the measurement model can be found in Table 2, with the residual of the cost function at termination being 9.98×10^{-5}.

$$cost = \sum_i (|g| - |X^s_{a,i}|)^2 \tag{13}$$

4.3 Gyroscope Calibration

The gyroscope cluster calibration coefficients were determined by minimising the cost function in Equation 14. In this equation $U_{a,i}$ are the rotation quaternions between consecutive acceleration vectors $X^s_{a,i}$ and $X^s_{a,i+1}$, using Equation 2, and $U_{a,i}$ are the rotation quaternions calculated by using Equation 5 to integrate the gyroscope readings, adjusted using Equation 10, between the stationary periods i and $i+1$.The minimisation was terminated with a residual of 3.61×10^{-6} and

the calculated coefficients for the measurement model can be found in Table 2.

$$cost = \sum_i (U_{a,i} - U_{g,i})^2 \tag{14}$$

4.4 Calibartion Paramaters

Triad	T_i	K_i	δ_i	σ_i
acceler-ometer	$\begin{bmatrix} 1 & -6.08\times10^{-3} & -6.04\times10^{-3} \\ 0 & 1 & 4.74\times10^{-3} \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 9.77 & 0 & 0 \\ 0 & 9.83 & 0 \\ 0 & 0 & 9.82 \end{bmatrix}$	$\begin{bmatrix} -9.79\times10^{-3} \\ -5.66\times10^{-4} \\ -1.30\times10^{-2} \end{bmatrix}$	$\begin{bmatrix} 6.05\times10^{-3} \\ 7.02\times10^{-3} \\ 6.57\times10^{-3} \end{bmatrix}$
gyroscope	$\begin{bmatrix} 1 & -3.70\times10^{-2} & -2.55\times10^{-2} \\ 1.49\times10^{-2} & 1 & -5.09\times10^{-3} \\ -5.91\times10^{-3} & -1.80\times10^{-2} & 1 \end{bmatrix}$	$\begin{bmatrix} 1.136 & 0 & 0 \\ 0 & 1.162 & 0 \\ 0 & 0 & 1.135 \end{bmatrix}$	$\begin{bmatrix} cl \\ cm \\ cn \end{bmatrix}$	$\begin{bmatrix} 9.23\times10^{-2} \\ 6.68\times10^{-2} \\ 7.24\times10^{-2} \end{bmatrix}$

Table 2: Values for the Measurment model in Section 4.2.4, found by optimising Equations 13 and 14.

5 Algorithm Performance Analysis

An analysis of the relative performance of the trajectory reconstruction algorithms, Deadrec.py and EKF.py, was then performed. For robust comparisons between the algorithms, the same data set, filtered across 5 bins, was used for both reconstructions. After the calibration period was completed, the sensor was moved through a predefined set of translations in the –Y, –X, Y, and X directions respectively to form a rectangle of size (dx,dy)=(42cm,30cm), with the start and end point at (x,y)=(0,0). During this motion the sensor was moved smoothly across a flat surface. The true trajectory of the sensor is plotted in Figures 5 and 6 as a dashed green line. Both algorithms utilised the initialisation method described in Section 3.2.1 but differed in the method by which attitude estimates were calculated. After the calibartion period was used to calculate the initial attitude and the gyroscope's zero offsets, the trajectory reconstruction was initialised at the end of the calibration period.

5.1 Raw Intergration Reconstruction (Deadrec.py)

The Deadrec.py script used the method detailed in Section 3.2.2 to intergrate the gyroscope measurements to calculate attitude estimates. The reconstructed X and Y velocities are plotted in Figure 3. During the initial stationary phase, A, the velocities begin to drift away from zero, marked with a green dashed line. This drift is due to errors developing in the attitude estimates. The error growth is caused by the random noise, and errors in the calculated zero offset, of the gyroscope. The errors in the attitude estimate change throughout the recostruction with a visibly differnt rate of acceleration during stationary phase B-E. The stability issues the errors in the attitude estimate cause can be seen in Figures 5 and 6. Whilst the translations are evident in the velocity plot, they are indistinguisible from the stationary phases in the trajectory plot. The final relative error, calculated as the displacment error at termination divided by the true length of the trajectory, is 0.88.

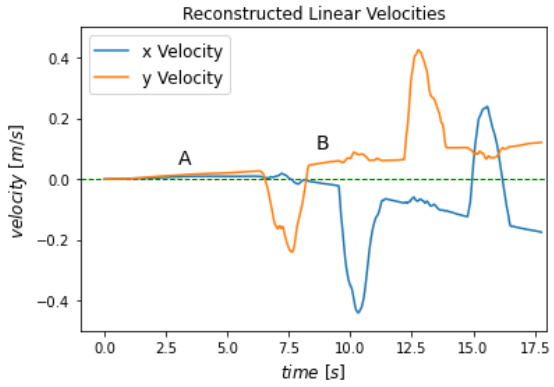

Figure 3: The velocities profile reconstructed using the Deadrec.py script. The dashed green line marks zero velocity.

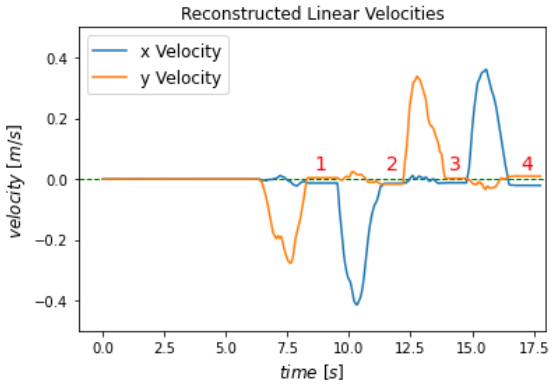

Figure 4: The velocity profile reconstructed using the EKF.py script. The dashed green line marks zero velocity. During the stationary phases labelled 1-4, the true velocity was zero.

5.2 Extended Kalman Filtering Reconstruction (EKF.py)

This EKF.py script utilised the accelerometer readings to calculate attitude estimates as described in Section 3.2.5, the weighting parameters of the accerometer estimate and gyroscope estimate were $\frac{1}{3}$ and $\frac{2}{3}$ respectivley. The condition for the EKF to activate was the magnitude of the linear accelerations in the navigation frame, predicted using the gyroscope estimate being less than $0.035m/s^2$. The stability gains are immediately obvious in the velocity profile plotted in Figure 4, with the velocity remaining zero throughout the initial stationary phase and the magnitude of the error during the other stationary phases (1-4) remaining below $0.03m/s$ (7.3% of the peak signal magnitude). Whilst errors do accumalate during the motion phases (where the EKF is not active), they remain small compared to the motion signal. This is demonstrated in Figure 6 where the displacements during the stationary phases 1-4, marked in red on the plot, are small compared to the signal displacements, marked in orange. The gain in stability and accuracy results in the reconstructed trajectory resembling the true trajectory in both scale and shape with a final relative error of 0.10.

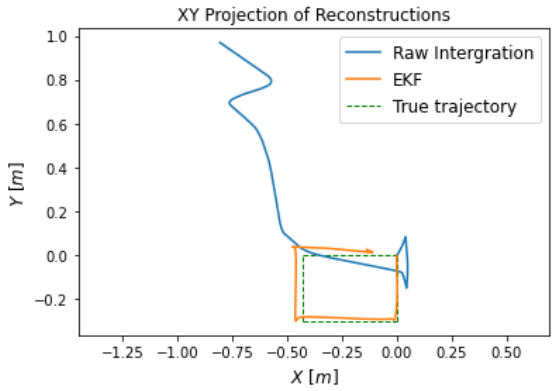

Figure 5: The trajectories reconstructed by the Deadrec.py (blue) and the EKF.py (orange) scripts. The true trajectory is plotted with a dashed green line.

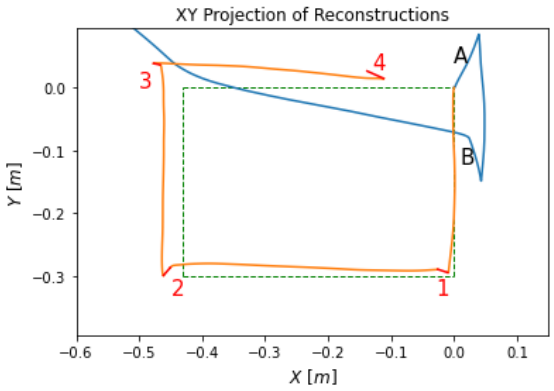

Figure 6: A subsection of Figure 4 with the stationary phases labelled for each reconstruction. Shares the same legend as Figure 4.

6 Concluding Remarks

The investifation of trajectory reconstruction methods identified the growth of errors in the attitude estimate as the limiting factor for the accuracy and stability of trajectory reconstructions. The major contributors to these errors are, the random noise distribution on the gyroscope readings, and the accuracy of the calibration parameters in the measurement model discussed in Section 3.2.4. Whilst the calibration accuracy can be increased by using more sophisticated calibration proceures the random noise distribution is inherent to the sensor itself. Consequently the sensor noise level will be a key determining factor in the accuracy achievable, and so should be a key focus of the design process.

However, it is possible to supress the drift in the Attitude estimates with the use of data fusion and an EKF as shown in Section 5.2. The demonstrated benefits of data fusion and an EKF will be a vital means by which the extend the period over which trajectories can be reconstructed accurately. The Aerobot will already be able to measure its alignment to the gravitational field using its accelerometer and the methods discussed in Section 3.2.5. Consequently, the system inherently has the ability to supress drift in the roll and pitch estimates. A design consideration for the Aerobot should be the possibility of including additional sensors, or leveraging information from already incorporated sensors, to provide additional attitude information (eg. orientation to the Sun or Earth). These additional measurements would provide heading information for the aerobot, which the gravity field is unable to do, and so drift would be constrained across all three axes. However, before any further instruments can be included in the design, a full cost-benefit analysis would need to be completed to decide if the weight, power, and complexity trade-off justify their inclusion.

An area in which further research could be conducted would be the inclusion of a motion model in the reconstruction algorithm. For the aerobot this could include features such as that the roll and pitch of the system is constrained with the cradle always hanging below the balloon envelope. As such the expected roll and pitch angles would fall as a sharply peaked distribution, Consequently a constraint could be placed on these these angles to ensure they follow this distribution, the exact nature of the restraint could only be determined by running CFD simulations of how the Aerobots geometry will respond to atmospheric disturbances. Another area to investigate would be if the problems, discussed in Section 1, with using a smoothing approach to solve the trajectory reconstruction problem can be overcome.

Finally, whilst the research presented here provides an indication of the factors that affect the ability to reconstruct the trajectory accurately, the growth of the errors in the reconstructed trajectory is highly dependent on both the nature of the motion and the ability of the reconstruction algorithm to supress the growth of these errors. Therefore, the minimum sensor specifications required to achieve the desired accuracy will not be known until the performance of the final EKF algorithm, that incorporates a motion model, is experimentally determined with either real or synthetic data sets that replicate the expected conditions on Venus. Thus there is significantly more work to be done before the time period over which the Aerobot's trajectory can be accurately reconstructed is known.

Appendices

A Mathematical Objects and Methods

A.1 Averaging Quaternions

The averaging of quaternions can be completed using the method outlined in **??**. The Method is described below:

1) Take the outer product of each quaternion with itself to form a set 4x4 matrices

$$Q_{mat,i} = q_i * q_i^T \tag{15}$$

2) Form a weighted sum of these 4x4 matricies:

$$Q_{mat} = \sum_i [q_i * q_i^T]\beta_i \tag{16}$$

3) Find the Eigen vectors and corresponding Eigen values of Q_{mat}:

$$Eig\_vec(Q_{mat}) \quad and \quad Eig\_val(Q_{mat}) \tag{17}$$

4) The average quaternion is the Eigen vector with the largest Eigen value.

A.2 Quaternion Integration definition

$$\Omega(X_{g,t}) = \begin{pmatrix} 0 & -X_l & -X_m & -X_n \\ X_l & 0 & Xn & -X_m \\ X_m & -Xn & 0 & X_l \\ X_n & Xm & -Xl & 0 \end{pmatrix} \tag{18}$$

Where l, m, and n are rotational rates about the x, y, and z axes respectiveley.

B Code Descriptions

Data_Import.py:
This script initialises a connection with the 33iot and imports the data in real time, writing it to a CSV file.
Inputs: Raw data from IMU.
Outputs: Table containing the time, accelerations, and angular velocities.

Filtering.py:
The Raw data file is imported and a new data frame of equal size is produced. The script then produces a moving average and writes these data values to the new data frame, which is then saved.
Inputs: Raw data Table.
Outputs: Filtered Data table.

Calibartion.py:

The data file is calibrated using the measurement model in Section 3.2.4.
Inputs: Filtered Data table.
Outputs: Calibrated Data table.

Deadrec.py:
Utilises the basic trajectory reconstruction methods outlined in Section 3.2, and follows the black path of the algorithm in Figure 1.
Inputs: Calibrated Data table.
Outputs: The reconstructed trajectory, velocity profile, and attitude.

EKF.py:
Utilises the accelerometer data to supress drift in the l and m rotations of the sensor, and follows the blue path of the Algorithm in Figure 1.
Inputs: Calibrated Data table.
Outputs: The reconstructed trajectory, velocity profile, and attitude.

Functions.py:
contains an array of functions utilised by multiple scripts.

References

[1] J. Blamont and R. Sagdeev, "The VEGA Mission," vol. 71, pp. 295–302, June 1984.

[2] M. Gilmore, P. Beauchamp, and et al, "2020 venus flagship mission study final report." https://www.lpi.usra.edu/vexag/reports/Venus-Flagship-Mission_FINAL.pdf, 2020. Accessed on 2021-03-21.

[3] C. Wilson and et al, "The 2010 european venus explorer (eve) mission proposal," 2011.

[4] K. Baines and et al, "New-frontiers (nf) class in-situ exploration of venus: The venus climate and geophysics mission concept," *Bulletin of the AAS*, vol. 53, 3 2021. https://baas.aas.org/pub/2021n4i346.

[5] R. Preston and et al, "Determination of Venus Winds by Ground-based Radio Tracking of the VEGA Balloons," vol. 231, pp. 1414–1416, Mar. 1986.

[6] G. P., *Principles of GNSS, Inertial, and Multisensor Integrated Navigation systems*. Boston, Mass. ; London: Artech House, 2008.

[7] M. kok, J. Hol, and T. Schon, "Using inertial sensors for position and orientation estimation," 2017.

[8] ST, "inemo inertial module: always-on 3d accelerometer and 3d gyroscope." https://www.st.com/resource/en/datasheet/lsm6ds3.pdf , Aug 2017. Accessed on 2021-04-11.

[9] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for imu calibration without external equipments," 2014.

[10] Arduino, "Arduino lsm6ds3 library." https://www.arduino.cc/en/Reference/Arduino_LSM6DS3, Dec 2019. Accessed on 2021-01-16.

[11] F. Markley, Y. C., C. J., and O. Y., "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, no. 30.4, pp. 1193–1197, 2007.

[12] M. Andrle and J. Crassidis, "Geometric integration of quaternions," 2013.

[13] Anon, "Scripts used for a035." https://github.com/Oxford-physmst/deadreckoning/upload/main, May 2021.

[14] L. J. and F. J., "Not fully overlapping allan variance and total variance for inertial sensor stochastic error analysis.," *IEEE Transactions on Instrumentation and Measurement*, no. 62.10, pp. 2659–2672, 2013.

[15] SciPy, "Optimization (scipy.optimize)." https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.htm, Apr 2021. Accessed on 2021-04-18.