Sensor Fusion in an iPhone 8

Jonathan Wheeler, *Stanford University*

ABSTRACT

This work seeks to create a software program that accurately tracks the position and orientation of an iPhone 8 using inertial sensors and GNSS sensors. Equations of motion that govern the state of a object in one, two, and three dimensions were derived. The noise performance of the iPhone 8 accelerometers and gyroscopes were characterized to be $60\,\mu\mathrm{g}/\sqrt{\mathrm{Hz}}$ and $8\,\mathrm{mdeg/s}/\sqrt{\mathrm{Hz}}$ respectively. Extended Kalman filters were derived for one-, two-, and three-dimensional cases. The one-dimensional case showed optimal performance (including during a simulated GNSS-outage). The two-dimensional case showed high accuracy despite indications that there was a bug in the code. The three-dimensional tests with an iPhone indicated that significant errors in the derivations of the equations of motion existed, causing the algorithm to diverge unless corrected by frequent GNSS updates. Corrective measures and further investigation are outlined.

INTRODUCTION

Navigation is an ancient discipline that is widely deployed in a variety of fields. Our first ancestors relied on navigation to find food, water, and shelter in their environment. Their primitive navigation protocol involved tracking their location relative to some fixed landmarks (e.g., walk one day towards the mountains in the distance, then turn towards the sea and walk for one day). Naturally, errors in their estimate of time, direction, and speed would cause errors in their final position. As our ancestors became more sophisticated, they traveled over regions where distinguishing landmarks were unavailable (e.g., open ocean) or dynamic (sand dunes). These travelers then developed the science of astronomy, where they used the stars to navigate in the absence of landmarks. Mapmaking also gave them opportunities to update their position estimates with a richer dataset of reference landmarks.

In recent years, micro-electromechanical systems (MEMS) became ubiquitous and low-cost. High-performing consumer-grade IMUs (such as those that appear in cell phones) cost a dollars, and provide users with estimates of their device's position and orientation. Such IMUs mitigate the need for constant availability of some reference direction (such as a mountain or the north star). However, due to the errors in an IMU sensor's offset, the errors in orientation and position accumulate without bounds unless some corrective action is taken to limit their growth. Thus, it is common for the estimates from an IMU filter to be aided by data from the GNSS, altimeters, magnetometers, cameras, radio beacons, and other reference sources.

The objective of this work is to develop a sensor-fusion algorithm using the Kalman filter and the data from an iPhone 8 IMU and GNSS sensor. The approach used is to test the filter on simulated data to ensure proper function, and then to observe the filter performance with sensor data acquired through the iPhone's CoreMotion and CoreLocation APIs.

Sensor Fusion of GNSS and IMU data in an iPhone 8

This project will entail the development of a Kalman filter which makes use of the following inputs programatically available in an iPhone 8:

- Estimates of GPS location (latitude, longitude, altitude)
- Accelerometer data from three axes in phone coordinate system (right, up, and out of the screen)
- Gyroscope data from three axes of phone coordinate system (yaw, pitch, and roll)

An iPhone also has an altimeter and a 3-axis magnetometer which can further be used for aiding, but will not be considered in this project.

Coordinate-frame conventions used in this work

For the purposes of this work, we establish a convention to distinguish the coordinates in the reference frame from the coordinates in the phone's frame. The global coordinate frame which we will select will be an orthonormal east-north-up (ENU) reference, and will be represented by the lower-case Cartesian coordinates x, y, and z. The coordinate frame for the inertial sensors (i.e., the accelerometers and gyroscopes) are in the phone's coordinate frame, which is documented in the Apple developer resources (see Fig. 1). Our convention will adopt upper-case Cartesian coordinates X, Y, and Z for this coordinate frame.

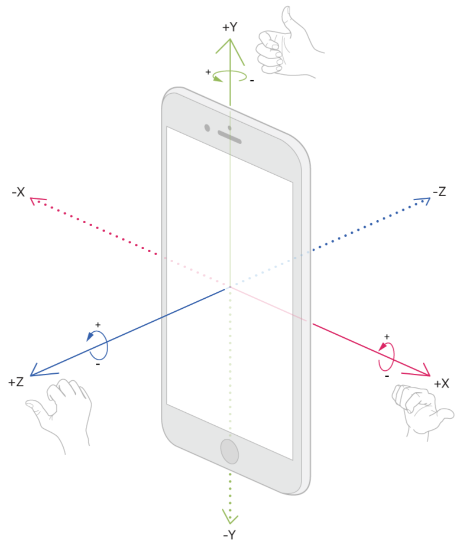


Figure 1: The coordinate system of the IMUs in an Apple product [1]

We can convert between these two rotation frames through a set of rotation matrices. We will use the yaw-pitch-roll Tait-Bryan angle convention often used to define the orientation of an aircraft (see Fig. 2). In this convention α represents yaw (i.e. anti-clockwise rotation about the z-axis), β represents pitch (i.e., anti-clockwise rotation about the y axis in the frame resulting from the yaw rotation), and γ represents roll (i.e., anti-clockwise rotation around the x axis in the frame resulting from the yaw and pitch rotations).

Thus, motion in the global coordinate frame can be transformed into the phone's inertial frame with the following rotation matrices

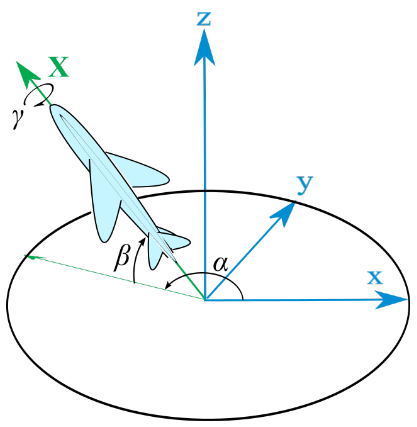


Figure 2: The Tait-Bryan angles that define an aircrafts orientation. Image adapted from [2]

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R(\alpha,\beta,\gamma)\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{1}$$

$$= R(0,0,\gamma)R(0,\beta,0)R(\alpha,0,0)\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2}$$

$$= R(0,0,\gamma)R(0,\beta,0)R(\alpha,0,0)\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3}$$

$$= \begin{pmatrix} \cos\gamma & 0 & \sin\gamma \\ 0 & 1 & 0 \\ -\sin\gamma & 0 & \cos\gamma \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & -\sin\beta & \cos\beta \end{pmatrix}\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{4}$$

$$= \begin{pmatrix} \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma & \cos\beta\sin\gamma \\ \sin\alpha\cos\beta & \cos\alpha\cos\beta & -\sin\beta \\ -\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{5}$$

The rotation from the XYZ frame to the xyz frame can be achieved by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R^T(\alpha,\beta,\gamma)\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{6}$$

owing to the fact that R is an orthogonal matrix, and that for any orthogonal matrix R, $R^{-1} = R^T$. In general, the transpose of a matrix is less computationally expensive to determine than its inverse.

As the body moves through space, it rotates, and its total orientation changes. The motion can be pictured as if the IMU is sitting on the axis of a slowly moving helicopter-like blade on top of the vehicle. That is, the vehicle moves linearly, but the IMU rotates at a constant rate. The rotation of the body about the $X-$, $Y-$, and $Z-$axes is detected by a MEMS chip with a gyroscope in each of those axes. The MEMS chip produces a reading for each axis (i.e., ω_X, ω_Y, and ω_Z) every sample period dt. The rotation in yaw, pitch, and roll directions ($\delta\alpha$, $\delta\beta$, and $\delta\gamma$ respectively) are given by

$$\begin{pmatrix} \delta\alpha \\ \delta\beta \\ \delta\gamma \end{pmatrix} = \begin{pmatrix} \omega_Z \\ \omega_Y \\ \omega_X \end{pmatrix} dt \tag{7}$$

The total rotation R_t at time t of the body after the gyroscope update is then

$$R_t = R(\delta\alpha, \delta\beta, \delta\gamma) R_{t-dt} \tag{8}$$

R_t is a 3×3 matrix, but only has three degrees of freedom (i.e., α, β, and γ). Rather than tracking all nine elements of R_t, we can calculate the values of α, β, and γ at every time step from R_t by exploiting several trigonometric identities of some of the elements.

$$\begin{aligned} R(\alpha,\beta,\gamma) &= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} && (9) \\ &= \begin{pmatrix} \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma & \cos\beta\sin\gamma \\ \sin\alpha\cos\beta & \cos\alpha\cos\beta & -\sin\beta \\ -\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\beta\cos\gamma \end{pmatrix} && (10) \end{aligned}$$

$$\begin{aligned} \frac{\sin\alpha}{\cos\alpha} &= \frac{r_{21}}{r_{22}} && (11) \\ -\sin\beta &= r_{23} && (12) \\ \frac{\sin\gamma}{\cos\gamma} &= \frac{r_{13}}{r_{33}} && (13) \end{aligned}$$

$$\begin{aligned} \alpha &= \operatorname{atan2}(r_{21}, r_{22}) && (14) \\ \beta &= \operatorname{asin}(-r_{23}) && (15) \\ \gamma &= \operatorname{atan}(r_{13}/r_{33}) && (16) \end{aligned}$$

The domain of β and γ are both $(-\pi/2, \pi/2)$, and the domain of α is $(-\pi, \pi]$. Therefore for software applications a function such as `arctan2` that can return values between $-\pi$ and π needs to be used for calculating α.

The system is degenerate when the device is vertical (i.e., Y is in the z direction), as there are infinitely many values of α and γ that will result in that transformation. At device configurations near this orientation, this method suffers from numerical instability and is generally suboptimal. These methods are the most stable for applications where the screen of the phone almost parallel to the surface of the earth.

In order to model a device free from any numerical instabilities, one might investigate using a more mathematically complex approach using quaternions [3, 4]. This approach was not adopted in this work due to the added mathematical complexity, especially in linearizing the state-evolution equations in the Kalman Filter described below.

PRINCIPLE OF A KALMAN FILTER

GNSS sensors enjoy the ability to estimate a position almost anywhere on earth's surface. Unfortunately, they have relatively slow bandwidth (1 Hz for an iPhone's GNSS sensor), and fairly low precision (between five and ten meters for an iPhone GNSS sensor). The accuracy can further be reduced when there is no direct line of sight between the GNSS receiver and at least four satellites.

On the other hand, inertial sensors enjoy low noise and high bandwidth. For example, the CoreMotion APIs on an iPhone allow data to be extracted as fast as 100 samples per second, and the noise of the iPhone accelerometers and gyroscopes are about $60\,\mu\mathrm{g}/\sqrt{\mathrm{Hz}}$ and $8\,\mathrm{mdeg/s}/\sqrt{\mathrm{Hz}}$ respectively (see a later section in this paper on sensor characterization). However, inertial sensors suffer from a relatively large offset that can drift over time. Error in the gyroscopes will cause errors in the estimate of

Characteristic	IMU only	GPS only	Sensor fusion
Bandwidth	fast	slow	fast
Noise performance	good	poor	good
Drift	unstable	stable	stable

Table 1: By using a carefully designed filter, a sensor-fusion technique provides state estimates using the best qualities of each constituent sensor.

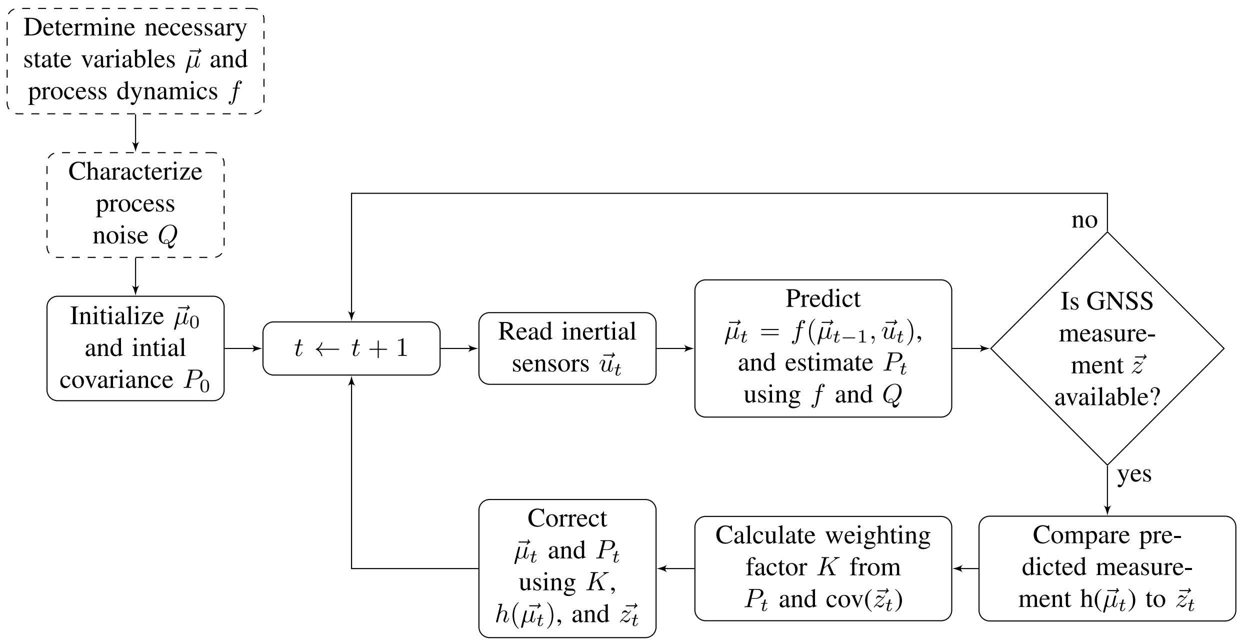

Determine necessary state variables $\vec{\mu}$ and process dynamics f
Characterize process noise Q
Initialize $\vec{\mu}_0$ and intial covariance P_0
$t \leftarrow t+1$
Read inertial sensors $\vec{u}_t$
Predict $\vec{\mu}_t = f(\vec{\mu}_{t-1}, \vec{u}_t)$, and estimate P_t using f and Q
no
Is GNSS measurement $\vec{z}$ available?
yes
Compare predicted measurement h($\vec{\mu}_t$) to $\vec{z}_t$
Calculate weighting factor K from P_t and cov($\vec{z}_t$)
Correct $\vec{\mu}_t$ and P_t using K, $h(\vec{\mu}_t)$, and $\vec{z}_t$

Figure 3: A flowchart illustrating the algorithm of the Kalman filter

the orientation of the IMU, and errors in the accelerometer will cause errors in the velocity estimate, which in turn cause errors in the position estimate. These errors grow as time t increases.

Take for an example an error Δa in the x-accelerometer. The error in the position estimate Δx can be calculated

$$\Delta x = \frac{1}{2}\Delta a t^2 \tag{17}$$

Take for example, an accelerometer whose offset has drifted by $100\,\mu g$. After one minute, the position error has grown to a modest $1.77\,\mathrm{m}$. However, after ten minutes, the error has grown to an unacceptably large $177\,\mathrm{m}$!

In order to bound this error, the predictions of an IMU are often aided by GNSS updates by means of some filter to produce a highly accurate estimate of the position and orientation at any time (see Table 1). Several varieties of filters have been developed [5, 6, 7]. For the purposes of this work, we will use an Extended Kalman filter (EKF), which allows almost-optimal fusing of various sensors while also handling nonlinear dynamics. When modeling a purely linear system, the EKF reduces to an ordinary Kalman Filter, which has been shown to have optimal performance [8]. Note that the EKF has some shortcomings, such as only being accurate to first-order expansions of the nonlinear equation. An unscented Kalman filter uses a slightly different approach to achieve third-order accuracy (in Taylor expansions) [9].

The algorithm of a Kalman filter is represented in Fig. 3. First, the user determines the state variables (both hidden and visible) $\vec{mu}$ that will be utilized by the filter. For this work, these variables are position, velocity, and orientation. The user also derives the equation $f(\vec{\mu}, \vec{u})$ which govern the system's evolution in time as well as the system's response to external stimulus (in our case, rotation rate and acceleration). Next, the user creates an estimate of how the uncertainty in the state variables grows with each time step and represents this with a square matrix Q with the same height as $\vec{\mu}$. The characterization of Q is discussed in a later section of this work.

The filter itself is initiated with initial estimates of the state $\vec{\mu}_0$ and state uncertainty in the covariance matrix P_0 where $p_{ij} = \mathrm{cov}(\mu_i, \mu_j)$. Unlike an IMU-only system, where the system evolution is unstable and highly sensitive to initial conditions, the EKF is less sensitive to errors in initial conditions due to the GPS updates that occur in the filter. Large values in P_0 indicate large uncertainty in the initial state estimate $\vec{\mu}_0$, and instruct the filter to rely heavily on the GPS updates to estimate $\vec{\mu}$.

The filter then begins its main loop. At every time step, the filter gathers measurements $\vec{u}$ from the IMU, and uses them to estimate how the state has evolved since the last time step. The filter also linearizes $f(\vec{\mu}, \vec{u})$ to calculate a Jacobian matrix F

$$F = \begin{pmatrix} \frac{\partial \mu_{1,t}}{\partial \mu_{1,t-1}} & \cdots & \frac{\partial \mu_{1,t}}{\partial \mu_{n,t-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mu_{n,t}}{\partial \mu_{1,t-1}} & \cdots & \frac{\partial \mu_{n,t}}{\partial \mu_{n,t-1}} \end{pmatrix} \tag{18}$$

The last step of the predict step is to update it's estimate of the covariance of the system. It does so by adding the process noise Q to the covariance of the previous timestep after it has passed through the state dynamics.

$$P_t = F P_{t-1} F^T + Q \tag{19}$$

If there are no new data from the GNSS sensor, then the calculations for this timestep are complete, and the filter awaits input from the next timestamp. However, if the GNSS has a new measurement $\vec{z}_t$, then the filter uses it to reduce the uncertainty of $\vec{\mu}_t$. Toward this end, the filter estimates what GNSS measurement it expects via the function $h(\vec{\mu})$. In this work, the GPS measurements $\vec{z}$ are in the same coordinate frame as the position state variables in μ. This allows $h(\vec{\mu}) = H\vec{\mu}$ where H is a $3 \times \dim(\vec{\mu})$ matrix with ones corresponding to the placements of x, y, and z in $\vec{\mu}$ and zeros elsewhere. The error between the actual measurement and the prediction of the measurement is given by the residual y

$$y = \vec{z}_t - H\vec{\mu}_t \tag{20}$$

In general, updating the state by the full residual would cause the system to overcorrect, since the GNSS data contains significant noise. To mitigate the amount of noise introduced to the system by correcting the residual, the residual is scaled by a gain factor K.

$$K = P_t H^T (\mathrm{cov}(\vec{z}) + H P_t H^T)^{-1} \tag{21}$$

If K is small, then the system relies heavily on the time evolution from the equations of motion. If K is large, then the filter relies heavily on the values of the GNSS update. The filter also uses the covariance in $\mathrm{cov}(\vec{z})$ to reduce the overall uncertainty in P. Note that the literature typically refers to $\mathrm{cov}(\vec{z})$ as R, but we shall refer to it henceforth as S in this text to avoid confusion with the rotation matrices (the literature makes use of P, Q, and R, thus S is the next letter in the alphabet to use).

Thus, the optimal correction equations are given by

$$\vec{\mu}_t \leftarrow \vec{\mu}_t + K(\vec{z}_t - H\vec{\mu}_t) \tag{22}$$
$$P_t \leftarrow (I - KH) P_t (I - KH)^T + KSK^T \tag{23}$$

After the correction, the timestep is incremented, and the filter awaits new data from the inertial sensors.

KALMAN FILTERING OF SIMULATED MOTION IN ONE DIMENSION

In order to build up our model, we took a number of incremental intermediate steps in order to catch any errors in the code early. A large number of simulation cases (at least ten) were performed, but only the few most interesting and pertinent studies are reported herein. First, a Kalman filter was programmed to model a simulated particle moving along one dimension. The moved a total of $10\,\mathrm{m}$ over the course of $60\,\mathrm{s}$, and particle's true position was given by

$$x(t) = 10\,\mathrm{m}\left(1 - \cos\left(\frac{2\pi t}{2\,\mathrm{min}}\right)\right)/2 \tag{24}$$

The IMU consisted of a single accelerometer oriented in the direction of motion with a sampling rate f_s of $10\,\mathrm{Hz}$ and a noise σ_a of $1\,\mathrm{mg}/\sqrt{\mathrm{Hz}}$. The GNSS receiver had a sampling rate of $1\,\mathrm{Hz}$ and an rms error σ_z of $1\,\mathrm{m}$.

The properties of the EKF were

$$\vec{\mu} = \begin{pmatrix} x & v \end{pmatrix}^T \tag{25}$$

$$Q = \begin{pmatrix} \sigma_a\sqrt{f_s}dt^2/2 & 0 \\ 0 & \sigma_a\sqrt{f_s}dt \end{pmatrix}^2 \tag{26}$$

$$u = a[t] \tag{27}$$

$$f(\vec{\mu}, \vec{u}) = \begin{pmatrix} 1 & dt \\ 0 & 1 \end{pmatrix}\vec{\mu} + \begin{pmatrix} dt^2/2 \\ dt \end{pmatrix} u \tag{28}$$

$$S = \sigma_z^2 \tag{29}$$

$$H = \begin{pmatrix} 1 & 0 \end{pmatrix} \tag{30}$$

$$P_0 = Q + R \tag{31}$$

Figure 4 shows the output of this filter given the simulated data described above. The outputs of the system guided by either the IMU or GNSS alone are also shown, as well as the true position of the particle. The width to the green trace illustrates the $\pm 1\sigma$ error bars (i.e., the filter estimates particle is 68%-likely to be within the green region). The 1-sigma uncertainty can estimated for all of the state variables at each timestep by taking the square root of the diagonals in the P vector.

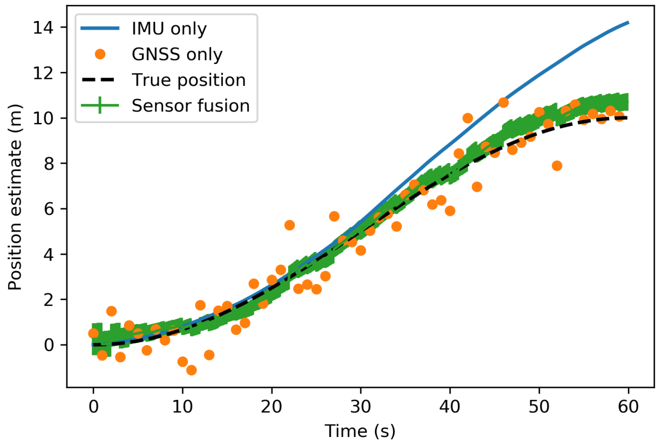


Figure 4: The results of the simulated particle accelerating in one dimension

The IMU-only system experiences large amounts of drift relative to the true position, while the GNSS-only system suffers from large amounts of noise and limited bandwidth.

The Kalman filter does a satisfactory job of merging the low drift of the GNSS data with the low noise of the accelerometer.

Next, we consider the same Kalman filter, but this time we simulate a GNSS outage from $t = 10\,\mathrm{s}$ until $t = 40\,\mathrm{s}$. The results of the Kalman filter are shown in Fig. 5

In Fig. 5, we see that the uncertainty of the Kalman-filter position state begins small, but when the GNSS updates become unavailable, the position estimate drifts relative to the true position, and the uncertainty grows. The Kalman filter's design

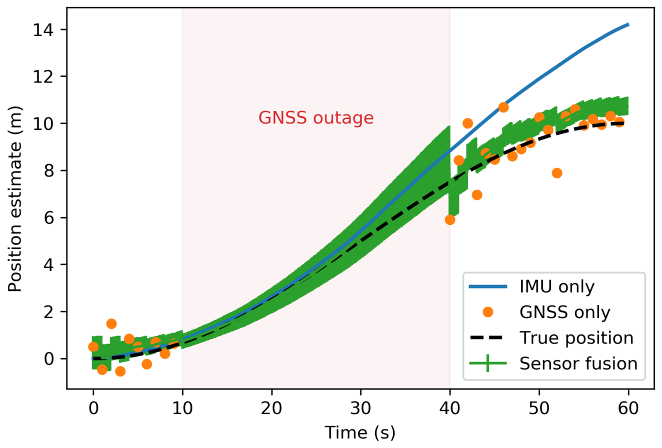


Figure 5: The results of the simulated particle accelerating in one dimension

allows it to realize that it has not received an update, and its error bars grow. In this iteration, the true position falls within the 1-sigma confidence interval during the entire outage, and when the GNSS data becomes available again, the filter quickly converges towards the correct value again. Towards the end of the simulation, the estimated position is a little larger than the true position, but this is due to both the accelerometer and the GNSS errors consistently pulling it in the positive direction.

KALMAN FILTERING OF SIMULATED MOTION IN TWO DIMENSIONS

To simulate motion in two dimensions, an IMU with two accelerometers (in the X and Y device axes) and one gyroscope (in the $z = Z$ axis) was modeled. The true position of the device was given by

$$\begin{pmatrix} x \\ y \end{pmatrix} = d_{max}\left(1 - \cos(2\pi t/T)\right)/2 \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \tag{32}$$

where d_{max} was $10\,\mathrm{m}$, θ was $\pi/4$, and T was $2\,\mathrm{min}$. The simulated IMU was subject to a rotation of $\Omega_Z = 0.1\pi\,\mathrm{rad/s}$, such that the axes of the accelerometers were constantly rotating with respect to the global reference frame.

The accelerations a_X and a_Y of the X- and Y-axis accelerometers and rotation ω_Z measured by the gyroscope were given by

$$\begin{pmatrix} a_X \\ a_Y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} + \begin{pmatrix} w_X \\ w_Y \end{pmatrix} \tag{33}$$

$$\omega_Z = \Omega_Z + w_Z \tag{34}$$

where $\ddot{x}$ and $\ddot{y}$ represent the second-order derivatives of x and y with respect to time (i.e., the true accelerations in those axes). The w terms represent noise added to the system, and are Gaussian random variables. Both w_X and w_Y have an rms error of $\sigma_a\sqrt{f_s}$ where σ_a is the characteristic noise of the accelerometers, and is assumed to be the same in the X and Y directions. w_Z has an rms error of $\sigma_g\sqrt{f_s}$ where σ_g is the characteristic noise of the gyroscope.

In this simulation, $f_s = 10\,\mathrm{Hz}$, $\sigma_a = 1\,\mathrm{mg}/\sqrt{\mathrm{Hz}}$, $\sigma_g = 1\,\mathrm{mrad/s}/\sqrt{\mathrm{Hz}}$. The GNSS receiver had a sampling rate of $1\,\mathrm{Hz}$ and an rms error σ_z of $1\,\mathrm{m}$ in both the x and y directions.

Note that these equations have neglected the Coriolis force and thus are slightly aphysical. As will be discussed in a further section, the omission of the Coriolis effect when filtering real sensor data will lead to errors. However, because no Coriolis forces are being added to the simulation, the Kalman filter will function well if its process equations do not include the Coriolis force.

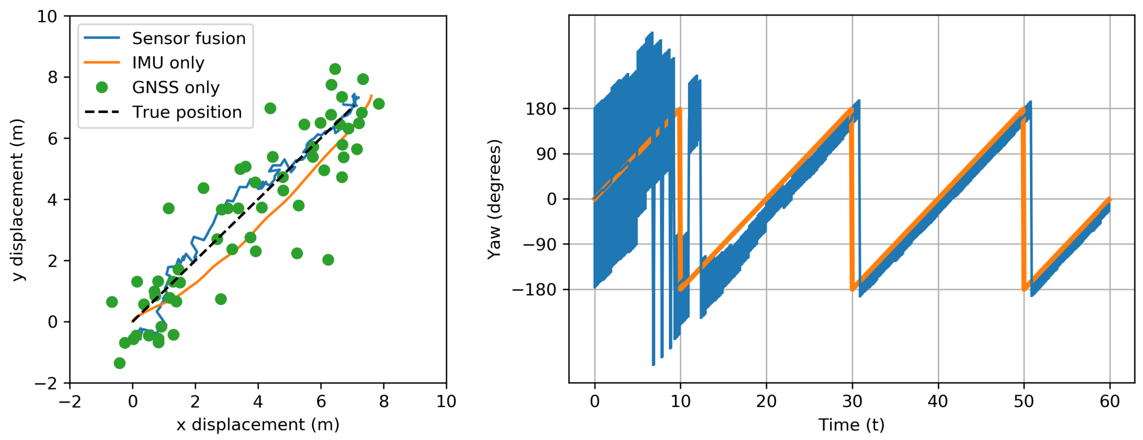


Figure 6: Left: A top-down view of the object as it propagates. The object moves from the bottom left to the top right. The estimated motion of the IMU without GNSS updates, the GNSS updates, the Kalman-filter output, and the true position are all plotted. Right: The yaw (clockwise rotation from the x-axis about the z-axis) is plotted with respect to time.

$$\vec{\mu} = \begin{pmatrix} x & y & \dot{x} & \dot{y} & \alpha \end{pmatrix}^T \tag{35}$$

$$Q = \begin{pmatrix} \sigma_a\sqrt{f_s}dt^2/2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_a\sqrt{f_s}dt^2/2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_a\sqrt{f_s}dt & 0 & 0 \\ 0 & 0 & 0 & \sigma_a\sqrt{f_s}dt & 0 \\ 0 & 0 & 0 & 0 & \sigma_g\sqrt{f_s}dt \end{pmatrix}^2 \tag{36}$$

$$\vec{u} = \begin{pmatrix} a_X & a_Y & \omega_Z \end{pmatrix}^T \tag{37}$$

$$f(\vec{\mu}, \vec{u}) = \begin{pmatrix} 1 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & dt & 0 \\ 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \vec{\mu} + \begin{pmatrix} \cos\alpha dt^2/2 & \sin\alpha dt^2 & 0 \\ -\sin\alpha dt^2/2 & \cos\alpha dt^2 & 0 \\ \cos\alpha dt & \sin\alpha dt & 0 \\ -\sin\alpha dt & \cos\alpha dt0 & 0 \\ 0 & 0 & dt \end{pmatrix} \vec{u} \tag{38}$$

$$F = \begin{pmatrix} 1 & 0 & dt & 0 & (-\sin\alpha a_X + \cos\alpha a_Y)\frac{dt^2}{2} \\ 0 & 1 & 0 & dt & (-\cos\alpha a_X + -\sin\alpha a_Y)\frac{dt^2}{2} \\ 0 & 0 & 1 & 0 & (-\sin\alpha a_X + \cos\alpha a_Y)dt \\ 0 & 0 & 0 & 1 & (-\cos\alpha a_X + -\sin\alpha a_Y)dt \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{39}$$

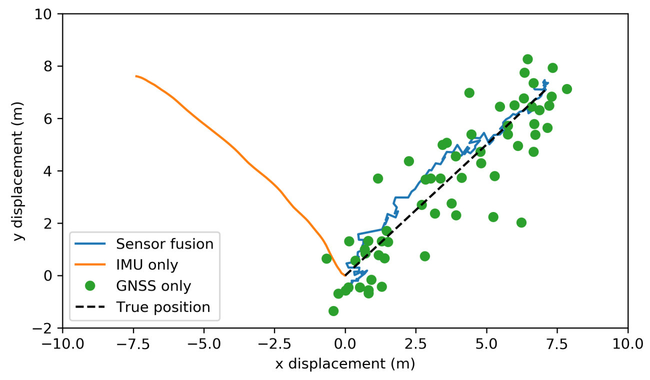
$$S = \begin{pmatrix} \sigma_z^2 & 0 \\ 0 & \sigma_z^2 \end{pmatrix} \tag{40}$$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{41}$$

$$P_0 = Q + R \tag{42}$$

Note that in this simulation, the element in the diagonal of P_0 corresponding to the uncertainty in yaw is set to π^2, indicating that we are very uncertain about our initial estimate in yaw.

Figure 6 illustrates the dynamics of the object as the simulation evolves. The motion of the particle as seen in a top-down view is illustrated on the left side of Fig. 6, where the object moves from the bottom-left corner to the top-right corner of the chart. The right side of the figure illustrates the yaw estimate in the sensor-fusion algorithm over time. At the beginning of the simulation, the algorithm is told that the estimate for the yaw is $0 \pm \pi$ radians. As the Kalman filter evolves with time, this



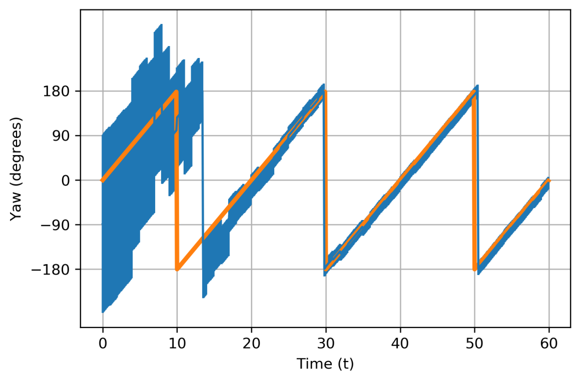


Figure 7: Left: A top-down view of the object as it propagates. In this simulation, the IMU received an incorrect initial estimate of the yaw. The true position of the object moves from the bottom center to the top right. The estimated motion of the IMU without GNSS updates, the GNSS updates, the Kalman-filter output, and the true position are all plotted. Right: The yaw (clockwise rotation from the x-axis about the z-axis) is plotted with respect to time.

estimate is refined and the error bars become quite small (only a few degrees). Even though the GNSS data does not contain any angular measurements (a magnetometer, on the other hand, would), the GNSS data can still reduce the uncertainty in yaw due to the off-diagonal terms in the F matrix. To understand how this is possible, consider the P matrix, which is influenced by the F matrix. The P matrix contains the covariances between different variables in the state. So while there may be an uncertainty cloud in position (the rms error of the position is given by the diagonal term), the left-part of the cloud may be correlated with positive errors in yaw, and the right-part of the cloud may be correlated with negative errors in yaw. If the GNSS update tells the sensor that the true position is in the left part of the cloud, the filter learns not only that the position estimate needs to be correct to the left, but the yaw estimate needs to be increased to a larger value.

In the process of writing this report, we noticed that the IMU-only position seemed to veer right of the true position, while the sensor-fusion position seemed to veer left. This could be indicative of the rotation matrices being backwards (i.e., a rotation matrix rotation clockwise when the user intends for it to rotate anticlockwise). Several attempts were made to remedy this apparent error, but with no success.

The simulation was repeated, but in this test the initial estimate of yaw was intentionally wrong by $-\pi/2$ (an error resulting in the object to be oriented $90°$ to the right). The results of this simulation are shown in Fig. 7.

Again, we see that the system seems to misinterpret the value of the yaw, such that a negative yaw induces a anticlockwise rotation about the Z axis, when it should be causing a clockwise rotation. Nevertheless, the Kalman filter still manages to cause the position and yaw estimates to converge to close of that of their true values, despite this apparent modeling error. The yaw error begins at $-90°$ and diminishes to a few degrees by the end of the simulation.

SIMULATING THE MOTION IN 3D SPACE

We did not have a chance to create a simulation of the motion in 3D space. Instead, we prioritized getting actual data for the final project. As will be discussed in a further section, our implementation of the 3D Kalman Filter seems to have errors, and thus one of the next steps is to perform the simulation in 3D space with known forces to debug and troubleshoot the problem.

CHARACTERIZING NOISE AND BIAS DRIFT WITH THE ALLAN-VARIANCE METHOD

The Apple API exposes the horizontal and vertical estimates for GNSS data. However, there is no publicly available documentation for the error performance of the IMU (nor does Apple even publish which IMU the iPhone uses!). In order to estimate the values to place into the Q matrix for the real iPhone data, the noise of each sensor needs to be characterized.

The IEEE defines a standard method of characterizing gyroscope and accelerometer errors [10]. This is done by placing the IMU at rest on some stable, vibration-free surface. The IMU collects data for a long period of time (the requisite length of

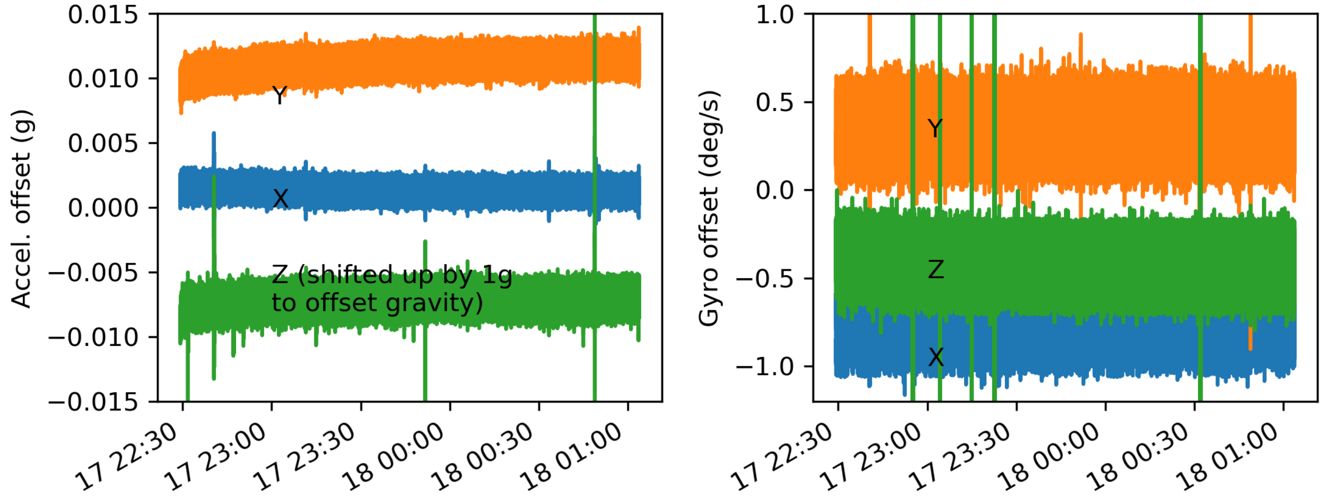


Figure 8: A 2.5-hour time trace for the sensor offsets over time. Note that the sensor drift is somewhat difficult to see as it is buried under the high noise when the sensor was sampled at 100 Hz

time is often a function of the bias-stability performance of the sensor; consumer-grade sensors need a few minutes to hours, military-grade sensors need weeks). Then, a statistical value called the Allan variance is calculated for a variety of integration times for each sensor. The Allan variance for a discrete signal $y[t]$ is given by

$$\mathrm{AVAR}(\tau) = \frac{1}{2}\left\langle (\bar{y}_{n+1} - \bar{y}_n)^2 \right\rangle \tag{43}$$

where

$$\bar{y}_n = \frac{dt}{\tau} \sum_{i=n\tau/dt}^{(n+1)\tau/dt} y[i] \tag{44}$$

and represents the average value for all of the data in the nth bin of data of bin-duration τ. It is also common to discuss the Allan deviation of a process, where the Allan deviation is the square root of the Allen variance (though it is common in the literature to refer to the a plot of the Allan deviation as an Allan-variance plot). The interpretation Allan deviation (ADev) for a given integration time τ is that the expected deviation between two consecutive averages of length τ is ADev(τ).

On a log-log plot of the ADev(τ) vs τ, the left hand side often is characterized by a long straight line with a slope (or exponent in linear plot) of -1/2. The intercept of this line with $\tau = 1\,\mathrm{s}$ indicates the noise of the sensor. The right side of the curve often gives rise to lines that are somewhat less linear with a slope of 0, +1/2, or +1. The standards defined in [10] give a greater description of the physical processes that drive the long-term offset fluctuations that give rise to each of these different slopes). In an advanced implementation of a Kalman filter, six additional state variables would be used to track the bias offsets of the six IMU sensors. While the gyro noise and accelerometer noise would determine the values of the Q-matrix for the orientation and velocity states respectively, the gyro drift and accelerometer drift (as estimated from the right-hand side of the log-log graph) would be used to determine the values of the Q-matrix associating with the rate of change of the IMU-sensor drifts.

The iPhone 8 to be characterized was placed on a flat surface and allowed to collect data for 2.5 hours. The time trace of the sensor offsets is shown in Fig. 8.

The Allan deviation plots for the data shown in Fig. 8 are shown in Fig. 9. The Allan-deviation plots reveal that accelerometers all have a noise performance of $60\,\mu\mathrm{g}/\sqrt{\mathrm{Hz}}$ or better, and the gyroscopes have a noise performance of $60\,\mathrm{mdeg}/\sqrt{\mathrm{Hz}}$ or better. The Allan deviation for the IMU sensors decreases with increasing integration time until τ is about $30\,\mathrm{s}$. This indicates that the offset does not drift a visible amount (relative to the sensor noise) under short time scales, but it does drift significantly (relative to the sensor noise) for long time scales. For short trajectories (a few minutes), the offset can be measured at the beginning of

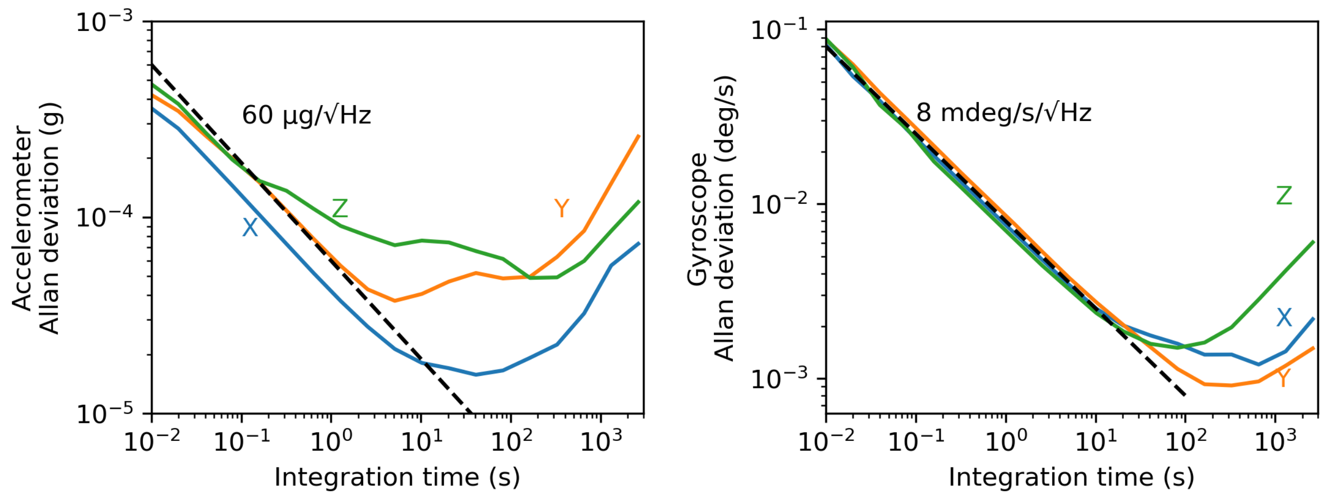


Figure 9: The Allan-deviation of the IMU sensors. The accelerometers all have a noise performance of $60\,\mu\text{g}/\sqrt{\text{Hz}}$ or better, and the gyroscopes have a noise performance of $60\,\text{mdeg}/\sqrt{\text{Hz}}$ or better.

the trajectory (before motion begins), and that offset can be used to null out any bias for the remainder of the tests. However, the drift is significantly large that the offset may vary over the course of several hours, so the offset from a test in the morning cannot in general be used to correct the bias error for a test in the evening.

KALMAN FILTERING OF AN IPHONE 8 IN THREE DIMENSIONS

We were not successful at producing a Kalman filter that accurately maintained the orientation of the iPhone 8 during field tests. The GNSS data was sufficient to cause the phone's location updates to converge somewhat close to the GNSS locations. Unfortunately, modeling error made the algorithm's estimate of the position unstable. Dozens of hours of effort were invested to isolate the problem, but without success.

The state of the most-recent Kalman filter implementation used in three dimensions is as follows

$$\vec{\mu} = (r_x, r_y, r_z, v_x, v_y, v_z, \alpha, \beta, \gamma)^T \tag{45}$$

where $\vec{r}$ corresponds to position and $\vec{v}$ corresponds to velocity in the global coordinate system.

$$\vec{u} = (a_X, a_Y, a_Z, \omega_Z, \omega_Y, \omega_Z)^T \tag{46}$$

where $\omega_Z, \omega_Y, and \omega_Z$ represent incremental changes in the yaw, pitch, and roll in the aircraft frame respectively.

$$Q = \left[I_9 \left(\sigma_a \sqrt{f_s} \frac{dt^2}{2}, \sigma_a \sqrt{f_s} \frac{dt^2}{2}, \sigma_a \sqrt{f_s} \frac{dt^2}{2}, \sigma_a \sqrt{f_s} dt, \sigma_a \sqrt{f_s} dt, \sigma_a \sqrt{f_s} dt, \sigma_g \sqrt{f_s} dt, \sigma_g \sqrt{f_s} dt \sigma_g \sqrt{f_s} dt \right)^T \right]^2 \tag{47}$$

Where the bandwidth f_s is $100\,\text{Hz}$ and the value of σ_a and σ_g are determined from the Allan-deviation characterization to be $60\,\mu\text{g}/\sqrt{\text{Hz}}$ and $8\,\text{mdeg}/\sqrt{\text{Hz}}$ respectively.

The value of the diagonal elements of S can be calculated at each timestep from the GNSS-sensor estimate of the horizontal and vertical accuracy provided by the iPhone API.

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{48}$$

$$P_0 = Q + S \tag{49}$$

where once again, we have set the diagonal element in P_0 corresponding to the yaw to be π^2 to represent our lack of confidence in the value's initialization.

The equations of motion f that govern the system are too complicated to be expressed concisely in a matrix form, and must be taken in steps. First, the orientation of the device is updated using the values from the previous state

$$R_t = R(\omega_Z, \omega_Y, \omega_Z)R(\alpha_{t-1}, \beta_{t-1}, \gamma_{t-1}) \tag{50}$$

$$\alpha_t = \text{atan2}(r_{21}, r_{22}) \tag{51}$$
$$\beta_t = \text{asin}(-r_{23}) \tag{52}$$
$$\gamma_t = \text{atan}(r_{13}/r_{33}) \tag{53}$$

Next, the acceleration in the global coordinate frame is calculated from a rotated version of the accelerometers

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = [R(\alpha_{t-1}, \beta_{t-1}, \gamma_{t-1})]^T \begin{pmatrix} a_X \\ a_Y \\ a_Z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \tag{54}$$

where the addition of the final term corrects for the gravitational field of earth.

Finally, the position and velocity vectors are updated

$$\vec{r}_t = \vec{r}_{t-1} + \vec{v}_{t-1}dt + \vec{a}dt^2/2 \tag{55}$$
$$\vec{v}_t = \vec{v}_{t-1} + \vec{a}dt \tag{56}$$

The linearization of the above steps is somewhat involved and very prone to human error when done by hand. In the present work, the terms of the linearization of the calculation of α_t, β_t, and γ_t are considered small to save us the work of linearizing the trigonometric functions of several terms. They are represented in the Jacobian matrix F as ϵ, but in the computational algorithm are treated as 0 for now. To calculate the dependence of the position and velocity measurements as a function of the error in the orientation, we begin by referencing Eq. 5 and calculating the transpose.

$$[R(\alpha, \beta, \gamma)]^T = \begin{pmatrix} \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\beta & -\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta & -\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & -\sin\beta & \cos\beta\cos\gamma \end{pmatrix} \tag{57}$$

Equation 54 becomes

$$a_x = (\cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma)a_X + (\sin\alpha\cos\beta)a_Y + (-\sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma)a_Z \tag{58}$$
$$a_y = (-\sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma)a_X + (\cos\alpha\cos\beta)a_Y + (-\cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma)a_Z \tag{59}$$
$$a_z = (\cos\beta\sin\gamma)a_X + (-\sin\beta)a_Y + (\cos\beta\cos\gamma)a_Z + g \tag{60}$$

The first-order derivatives with respect to the Tait-Bryan angles are

$$\frac{\partial}{\partial\alpha}\begin{pmatrix}a_x\\a_y\\a_z\end{pmatrix} = \begin{pmatrix}-\sin\alpha\cos\gamma-\cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta & -\cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma\\ -\cos\alpha\cos\gamma+\sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta & \sin\alpha\sin\beta\cos\gamma+\cos\alpha\sin\gamma\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}a_X\\a_Y\\a_Z\end{pmatrix} \tag{61}$$

The row of zeros indicates that the global acceleration in the z direction is not sensitive to a_Z for errors in yaw

$$\frac{\partial}{\partial\beta}\begin{pmatrix}a_x\\a_y\\a_z\end{pmatrix} = \begin{pmatrix}-\sin\alpha\cos\beta\sin\gamma & -\sin\alpha\sin\beta & -\sin\alpha\cos\beta\cos\gamma\\ -\cos\alpha\cos\beta\sin\gamma & -\cos\alpha\sin\beta & -\cos\alpha\cos\beta\cos\gamma\\ -\sin\beta\sin\gamma & -\cos\beta & -\sin\beta\cos\gamma\end{pmatrix}\begin{pmatrix}a_X\\a_Y\\a_Z\end{pmatrix} \tag{62}$$

$$\frac{\partial}{\partial\gamma}\begin{pmatrix}a_x\\a_y\\a_z\end{pmatrix} = \begin{pmatrix}-\cos\alpha\sin\gamma-\sin\alpha\sin\beta\cos\gamma & 0 & \sin\alpha\sin\beta\sin\gamma-\cos\alpha\cos\gamma\\ \sin\alpha\sin\gamma-\cos\alpha\sin\beta\cos\gamma & 0 & \cos\alpha\sin\beta\sin\gamma+\sin\alpha\cos\gamma\\ \cos\beta\cos\gamma & 0 & -\cos\beta\sin\alpha\end{pmatrix}\begin{pmatrix}a_X\\a_Y\\a_Z\end{pmatrix} \tag{63}$$

The column of zeros signifies the error in global acceleration does not depend on the a_Y accelerometer if the roll is varied.

Thus, the linearized vector F is

$$F = \begin{pmatrix}1 & 0 & 0 & dt & 0 & 0 & \frac{\partial a_x}{\partial\alpha}\frac{dt^2}{2} & \frac{\partial a_x}{\partial\beta}\frac{dt^2}{2} & \frac{\partial a_x}{\partial\gamma}\frac{dt^2}{2}\\ 0 & 1 & 0 & 0 & dt & 0 & \frac{\partial a_y}{\partial\alpha}\frac{dt^2}{2} & \frac{\partial a_y}{\partial\beta}\frac{dt^2}{2} & \frac{\partial a_y}{\partial\gamma}\frac{dt^2}{2}\\ 0 & 0 & 1 & 0 & 0 & dt & \frac{\partial a_z}{\partial\alpha}\frac{dt^2}{2} & \frac{\partial a_z}{\partial\beta}\frac{dt^2}{2} & \frac{\partial a_z}{\partial\gamma}\frac{dt^2}{2}\\ 0 & 0 & 0 & 1 & 0 & 0 & \frac{\partial a_x}{\partial\alpha}dt & \frac{\partial a_x}{\partial\beta}dt & \frac{\partial a_x}{\partial\gamma}dt\\ 0 & 0 & 0 & 0 & 1 & 0 & \frac{\partial a_y}{\partial\alpha}dt & \frac{\partial a_y}{\partial\beta}dt & \frac{\partial a_y}{\partial\gamma}dt\\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{\partial a_z}{\partial\alpha}dt & \frac{\partial a_z}{\partial\beta}dt & \frac{\partial a_z}{\partial\gamma}dt\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \epsilon_1 & \epsilon_2\\ 0 & 0 & 0 & 0 & 0 & 0 & \epsilon_4 & 1 & \epsilon_4\\ 0 & 0 & 0 & 0 & 0 & 0 & \epsilon_5 & \epsilon_6 & 1\end{pmatrix} \tag{64}$$

In software implementations, because no analytic solution to the six values of ϵ was derived, they were treated as zero. The Kalman filter was, however, programmed with a second update step (in addition to the GNSS update). In this update step, which only occurred when the magnitude of the total acceleration was close to g (i.e., $a_X^2 + a_Y^2 + a_Z^2 \approx g^2$), the Kalman filter assumed that the device was not accelerating, and that the direction of the total acceleration was equivalent to the direction of the gravity vector. This allowed for an update to the pitch and the roll of the device. For this correction,

$$h(\vec{\mu}) = R(\alpha,\beta,\gamma)(0,0,-g) \tag{65}$$

which, when linearized is

$$H = g\begin{pmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin\beta\sin\gamma & -\cos\beta\cos\gamma\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\sin\beta\cos\gamma & -\cos\beta\sin\gamma\end{pmatrix}\vec{\mu} \tag{66}$$

The value of z was $(a_X, a_Y, a_Z)^T$, and the value of S was estimated by the magnitude of $(a_X^2 + a_Y^2 + a_Z^2)/g^2 - 1$

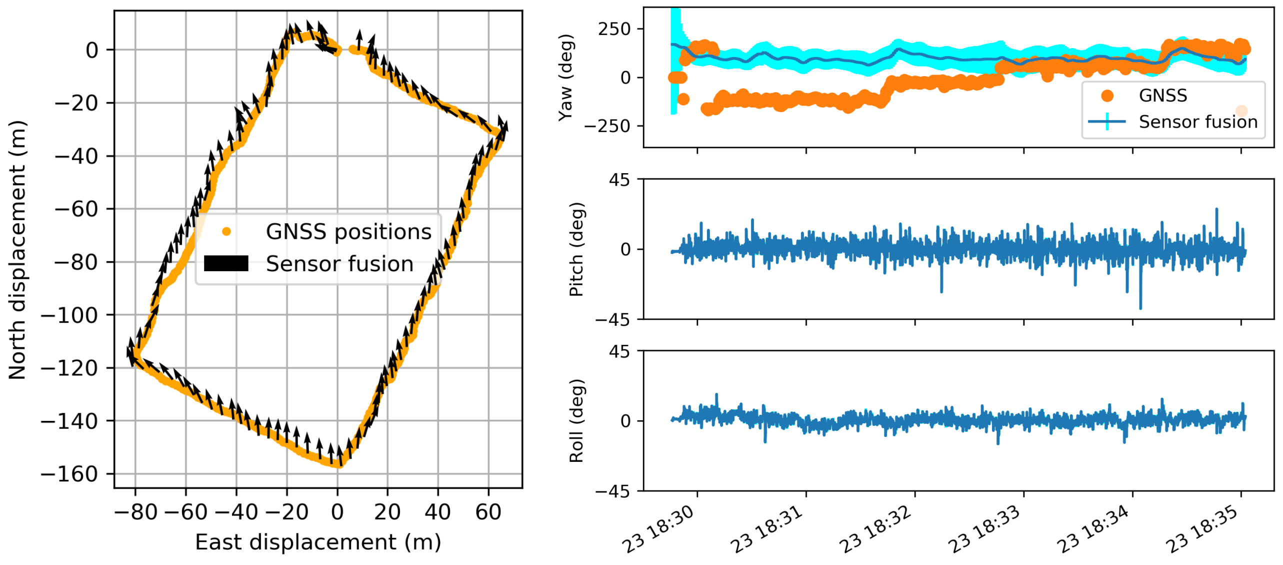


Figure 10: Left: a birds-eye view of the trajectory taken during the field test. The path started near the top of the curve, and followed the rectangular path counter-clockwise. The orange dots represent position estimates of the GNSS sensor, and the black arrows represent locations and orientations of the phone as estimated by the Kalman filter. Right: Three time traces of the Kalman filter's estimate of yaw, pitch, and roll. The orange dots in the top chart represent the heading (yaw) estimate based on the difference between subsequent GNSS updates. The error bars are plotted in cyan.

To test the filter, an iPhone was placed on the seat of a bicycle so that it's Y direction faced towards the front of the bicycle. The X direction was facing to the right of the bicycle, and the Z direction was pointing straight up into the sky. The bicycle was walked at a constant speed clockwise around one city block. The results of this test are shown in Fig. 10.

The Kalman filter proved to be very unstable. When the values of the Q matrix were small (corresponding to low-noise in the IMU sensors), the algorithm ignored updates from the GNSS sensor. The position estimates quickly diverged from the local neighborhood and the phone's estimated position left the bay area and departed into outer space. This was likely due to some errors in how the Kalman filter corrects for orientation.

In order to get the sensor-fusion position estimates to be somewhat close to the GNSS positions, the Q matrix had to be updated to reflect large noise in the accelerometers. Thus, the algorithm trusted the GNSS updates more than the IMU sensors, and the error converged to the sidewalk upon which the bicycle was moving.

Even then, the estimates for the heading of the device did not seem to track the ground truth. The phone was always positioned in the direction of travel, so the arrows in Fig. 10 should have been pointing parallel to the direction of travel. Instead, the arrows in Fig. 10 seem to point mostly north, and do not seem to be obviously correlated with the path direction or any other test quantity.

The pitch and the roll, on the other hand, converged nicely. However, this is because these two quantities were determined by a feedback correction mechanisms (using an calculation that predicts these values through $z - h(\vec{\mu})$ instead of $f(\vec{\mu}, \vec{u})$). There are no visible error bars in Fig. 10, and the "noise" in pitch represent bumps in the aging sidewalks where the test was performed, and the drift in the roll represent the fact that the bicycle leaned a little to the right and a little to the left as it was pushed around the block.

As mentioned previously, the instability of this filter and its overall failure to correctly calculate the heading (yaw) correctly are likely due to a failure to derive and linearize the equations of motion correctly. The linearized F matrix contains no fewer than 29 non-zero terms. In principle, all 29 of these terms need to be calculated correctly in order for the filter to perform as required. Furthermore, to save on mathematical complexity, the calculation of the pitch and roll was performed by comparing the accelerometer vector to the gravity vector in a feedback step, rather than propagating the uncertainties a predict step. As has been seen in previous sections, most of the code used in the Kalman filter software implementation functions correctly, but likely a few signs were flipped or a derivative was misapplied in the calculation of F which caused the system to become unstable. Further debugging is likely to uncover the source of the problem, and then the sensor would perform optimally, yielding errors of less than a meter without having to artificially increase the values of Q to minimize IMU drift.

CONCLUSION

We have demonstrated EKFs suitable for tracking the motion and orientation of objects in one, two and three dimensions. The EKFs make use of IMU sensor data and are corrected periodically by measurements from a GNSS sensor. These EKFs are also tested using the IMU and GNSS chips in an iPhone 8 using trajectories that a typical pedestrian might experience.

In principle, the construction of an EKF is straightforward, but prone to human errors. The equations that govern the laws of motion involve significant amounts of trigonometry, and deriving the Jacobian of the state evolution function by hand includes dozens of terms. If any of these terms is incorrect or takes the wrong sign, the EKF is no longer stable and causes the estimation of the state to quickly diverge from the true state.

Furthermore, the approach used to track the device orientation makes use of a yaw-pitch-roll coordinate frame which becomes degenerate (due to gimbal lock) when the device is either vertically up or down. This causes large numerical instabilities when the device is in these orientations (e.g., the user is taking a picture of some object at eye-level, or the device is in the user's pocket). An alternative solution implements quaternions, though this comes at the added challenge of linearizing the rotation operations between the IMU's reference frame and the global reference frame.

Further Work

The performance of these Kalman filters could further be enhanced by accounting for other forces (such as centripetal acceleration and the Coriolis force) in the state-evolution equations in the Kalman filter.

Each of the gyroscopes and accelerometers has some offset that varies with time. A high-performance Kalman filter would include six additional terms in its state model to estimate the gyroscope and accelerometer biases. While the bias of any given sensor can be modeled somewhat using a knowledge of the temperature and magnetic field at the sensor, some of the offset drift still is difficult to develop a physical model for, but the expectation value of its rate of its drift over time can be characterized experimentally and used to define the process noise for its associated Q-matrix values.

The iPhone API does not expose the pseudorange measurements used to calculate the GNSS latitude, longitude, and altitude. It merely provides a latitude, longitude, altitude, horizontal accuracy, and vertical accuracy. In principle, an EKF would have superior performance (even in urban environments where only one or two satellites may be available) by using $h(\vec{\mu})$ to calculate the expected pseudoranges to each satellite, and comparing $h(\vec{\mu})$ to the measured pseudoranges $\vec{z}$. This algorithm is similar to the gradient-descent algorithm already used to estimate the GNSS position, except that in the absence of IMU data, four pseudoranges are required to estimate x, y, z, and t. The data in an EKF can ease this 4-satellite constraint. In urban environments, this would help bound the vertical error, which already is somewhat unstable due to the numerical difficulty of canceling out the earth's gravitational bias on the z-accelerometer.

REFERENCES

[1] Apple Inc., "Getting raw gyroscope events." https://developer.apple.com/documentation/coremotion/getting_raw_gyroscope_events.

[2] Wikimedia Foundation, "Euler angles." https://en.wikipedia.org/wiki/Euler_angles.

[3] E. Kraft, "A quaternion-based unscented kalman filter for orientation tracking," *Proceedings of the Sixth International Conference of Information Fusion*, 2003.

[4] L. Wang, Z. Zhang, and P. Sun, "Quaternion-based kalman filter for ahrs using an adaptive-step gradient descent algorithm," *International Journal of Advanced Robotic Systems*, 2015.

[5] R. Mahony, T. Hamel, and J.-M. Pfimlin, "Complementary filter design on the special orthogonal group $so(3)$," *44th IEEE Conference on Decision and Control*, 2005.

[6] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, “Estimation of imu and marg orientation using a gradient descent algorithm,” *2011 IEEE International Conference in Rehabilitation Robotics*, 2011.

[7] G. Baldwin, R. Mahony, J. Trumpf, T. Hamel, and T. Cheviron, “Complementary filter design on the special euclidean group so(3),” *2007 European Control Confference*, 2007.

[8] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[9] E. Wan and R. V. D. Merwe, “The unscented kalman filter for nonlinear estimation,” *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000.

[10] IEEE, “Ieee standard specification format guide and test procedure for single-axis interferometric fiber optic gyroscopes,” 2008.