ARL-TR-9798 • Sep 2023

DEVCOM ARMY RESEARCH LABORATORY

Sources of Error in GPS-Denied Odometry

by Justin Brody, Sarah Leung, Jared Shamwell, and Daniel Donavanik

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9798 ● SEP 2023

Sources of Error in GPS-Denied Odometry

by Justin Brody
Franklin and Marshall College

Sarah Leung and Jared Shamwell
Draper Laboratory

Daniel Donavanik
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2023	Technical Report	October 2020-October 2022

4. TITLE AND SUBTITLE
Sources of Error in GPS-Denied Odometry

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Justin Brody, Sarah Leung, Jared Shamwell, and Daniel Donavanik

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLA-PB
Adelphi Laboratory Center, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9798

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
primary author's email: <justin.d.brody.ctr@army.mil>, <justin.brody@fandm.edu>.

14. ABSTRACT
As GPS-denied navigation becomes increasingly important, it is crucial to understand how errors in GPS-odometry arise. To that end, we model several scenarios of people and vehicles performing inertial odometry with various grades of intertial measurement units (IMUs). We compute the error in the odometry attributable to noise in various IMUs and to clock noise, ultimately concluding that, even with high end IMUs, error propagates significantly.

15. SUBJECT TERMS
inertial odometry; GPS-denied localization; IMU noise; Kalman filtering; Network, Cyber and Computational Sciences

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Daniel D Donavanik
Unclassified	Unclassified	Unclassified	UU	35	19b. TELEPHONE NUMBER (Include area code) 301-394-0728

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Soldiers and the machines that support them have a critical need to know where they are at any given moment. In the presence of an accurate global positioning system (GPS) signal, this *localization* problem can be solved with a high degree of precision. Unfortunately, the reliable presence of such a signal cannot be taken for granted in all scenarios as GPS signals are notoriously easy to jam or even spoof. Thus, there is a critical military need to develop the capacity to accurately determine position in the absence of GPS signals.

A number of different technologies have been proposed to address this need; this report is an attempt to begin a measurement of the performance characteristics of each. In particular, we present simulation results that examine the error rates associated with several algorithms. This allows us to make accurate estimates of the relative contribution to the error of various components.

Our primary focus is on the loosely coupled integration of a strapdown inertial navigation system[1] with visual odometry algorithms fused via Kalman filtering. In subsequent sections, we will present a detailed explanation of our methodology and findings.

Our study looked at two basic scenarios: a person walking at 2 m/s and a car driving at 10 m/s. Each scenario is modelled as a random walk. For each case, the ground truth of the simulated movement was compared to the results of

- Position estimation based on pure inertial measurement unit (IMU) integration.
- Position estimation based on pure visual odometry; that is, estimating position and attitude changes by comparing sequential camera images.
- Position estimation based on a Kalman-filter-based fusion of the above estimates.

2. Methodology

Our results were obtained by averaging the results of running 10 simulations. Each simulation comprises the following components:

- Ground truth trajectory generation
- Noise modelling
- Estimated trajectory generation with simulated IMU reckoning
- Estimated trajectory generation with simulated visual odometry
- Estimated trajectory generation with Kalman filtering

2.1 Notation and Preliminaries

This section defines some basic terms and notation.

When working with a navigational system, there are a number of different coordinate frames one can work in. Usually, a given calculation will be more natural in one coordinate frame; the results of this computation will then need to be projected into the appropriate coordinate frame. Such frame translations make up a great deal of the computational complexity of working with navigation.

We thus begin by briefly defining the relevant reference frames (much more detail is available in the literature; for example see Titterton and Weston[1]). Our main reference frames are the following:

- **Inertial**. This is a kind of global frame, defined as a reference frame is not itself accelerating.
- **Latitude-Longitude-Altitude** (abbreviated as "LLA"). This frame measures all forces relative to spherical coordinates on the Earth.
- **North-East-Down** (abbreviated as "NED"). This is a local reference frame that measures forces in terms of their components pointing North, East, and Down. It is generally used to make calculations relative to the system's current position.
- **Body**. This reference frame, centered at the body of the navigational system, defines three orthonormal axes that align with the system.
- **Sensor**. This is a reference frame similar to the body frame, but is specifically defined in terms of a particular sensor on the body of the system.

Throughout the rest of this section, we adopt the following mathematical conventions:

- $\mathbf{R}$ represents the set of real numbers.
- $\mathbf{Z}$ represents the set of integers.
- $M(a, b)$ represents the ring of all $a \times b$ real-valued matrices.
- $\mathcal{N}(\mu, \sigma^2)$ represents the Gaussian distribution with mean μ and standard deviation σ. If we write $X \sim \mathcal{N}(\mu, \sigma^2)$, then we mean that X is a random variable which is drawn from that distribution. Similarly, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ indicates a multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

2.2 Ground Truth Trajectory Generation

For each type of simulation (pedestrian or automobile), we first generated the ground-truth IMU data. Our basic goal was to generate reasonable approximations of what IMU readings for a walking pedestrian or driving car might be. In particular, in both situations we simulated movement at a regular speed using the trajectory generation mechanisms of Aceinna's *GNSS inertial navigation, sensor fusion simulator*.[2] This in turn emulates a proportional-integral-derivative (PID) controller[3] to generate smooth data given a path specified in terms of position and attitude changes.

Our input to the PID algorithm consists of the following:

- An initial velocity v_0 (in meters per second), an initial position p_0 (in latitude-longitude-altitude coordinates), and an initial attitude α_0 (expressed as roll-pitch-yaw using degrees in NED coordinates). We used $v_0 = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T$ for simulating pedestrians and $v_0 = \begin{bmatrix} 10 & 0 & 0 \end{bmatrix}^T$ for simulating a vehicle. For all scenarios, we set $p_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and $\alpha_0 = \begin{bmatrix} -180 & 0 & -90 \end{bmatrix}^T$ (the perhaps unexpected value for α_0 is to translate from the sensor frame to the NED frame) .
- A total duration T (in seconds). Our simulations were each 1 h, so we set $T = 3600$.

- An IMU sample rate fs that indicated the frequency of IMU data, measured in hertz. We used $fs = 100$. In what follows, we will use dt to denote $\frac{1}{fs}$; that is, dt is the number of seconds between IMU samples.
- A command step-size cdt, measured in seconds. The algorithm produces new position, velocity, and attitude information based on updates of cdt seconds. We set $cdt = 1$. We require that cdt be an integral multiple of dt.
- Step-wise velocity and attitude white noise: $N_v \sim \mathcal{N}(0, \Sigma_p), N_\alpha \sim \mathcal{N}(0, \Sigma_\alpha)$ for diagonal matrices $\Sigma_p, \Sigma_\alpha \in M(3, 3)$. Also acceleration and attitude change white noise: $N_{\dot{v}} \sim \mathcal{N}(0, \Sigma_{\dot{v}}), N_{\dot{\alpha}} \sim \mathcal{N}(0, \Sigma_{\dot{\alpha}})$ for diagonal $\Sigma_{\dot{v}}, \Sigma_{\dot{\alpha}} \in M(3, 3)$
- Accelerometer and gyroscope noise variables $N_A \sim \mathcal{N}(0, \Sigma_A), N_G \sim \mathcal{N}(0, \Sigma_G)$

Finally, let $\mathbf{T} = \{\, k(dt) : k \in \mathbf{Z}, 0 \leq k(dt) \leq T \,\}$ so that $\mathbf{T}$ is our set of time indices. We also let $\mathbf{T}^+ = \mathbf{T} \setminus \{\, 0 \,\}$; these represent the time indices after time 0. Given these, our basic algorithm is as follows:

1. Generate velocity and attitude. For $t \in \mathbf{T}$, set $v_t = v_0 + N_v$ and $\alpha_t = \alpha_0 + N_\alpha$
2. Generate derivatives of the above. For $t \in \mathbf{T}$
$$\frac{dv_{t+1}}{dt} = \frac{v_{t+1} - v_t}{dt}$$
$$d\alpha_{t+1} \sim N_{\dot{\alpha}}$$
3. Pass $\{\, dv_t : t \in \mathbf{T} \,\}$ and $\{\, d\alpha_t : t \in \mathbf{T} \,\}$ to GNSS-INS-SIM[2]; this will return reference accelerometers, gyroscopes, positions, velocities, and attitudes as well as initial position, velocity, and attitude.

Then our algorithm outputs the following sets:

- Sets of time-indexed reference data for accelerometers and gyroscopes: $\mathbf{A_R} := \{\, A_t : t \in \mathbf{T} \,\}, \mathbf{G_R} := \{\, G_t : t \in \mathbf{T} \,\}$. These correspond to the ground truth IMU data.

- Sets of time-indexed readings for accelerometers and gyroscopes: $\mathbf{A_N} := \{ A_t^N : t \in \mathbf{T} \}, \mathbf{G_N} := \{ G_t^N : t \in \mathbf{T} \}$, where, for $t \in \mathbf{T}$

$$A_t^N = A_t + N_A$$
$$G_t^N = G_t + N_G$$

  Thus $\mathbf{A_N}, \mathbf{G_N}$ represent the noised IMU readings available to the navigation algorithm.

- Sets of time-indexed positions $\mathbf{P}^0 := \{ p_t^0 : t \in \mathbf{T}^+ \}$, velocities $\mathbf{V} := \{ v_t : t \in \mathbf{T}^+ \}$ and attitudes $\mathbf{O} := \{ \alpha_t : t \in T^+ \}$. These are chosen so that they are consistent with the ground-truth data $\mathbf{A_R}, \mathbf{G_R}$.

A graph of the generated $\mathbf{A_N}, \mathbf{G_N}$ is given in Fig. 1.

There is a somewhat significant discrepancy between $\mathbf{P}^0$ as returned by the GNSS-INS-SIM software and the result of mechanizing the IMU data $\mathbf{V}, \mathbf{O}$ in Scorpion. For consistency, we therefore let $\mathbf{P}$ be the result of mechanizing $\mathbf{V}, \mathbf{O}$ with Scorpion—this will function as our ground-truth for position.

2.3 Noise Modelling

In order to simulate the behavior of our localization algorithms, we must create noise models that reflect the data which will be available to the algorithms. In this section we describe the noise models used for various components of the system.

2.4 IMU Noise

As the name implies, an "inertial measurement unit" is simply a device for measuring inertial forces. In the context of navigation and odometry, the term refers usually to accelerometers that measure acceleration (or, more precisely, the "specific force" $\mathbf{f} = \mathbf{a} - \mathbf{g}$ where $\mathbf{a}$ is the inertial acceleration and $\mathbf{g}$ is the local gravitational force) and gyroscopes which measure the change in orientation (attitude) of a system (or less frequently, the absolute orientation of the system).

Both terms cover a range of technologies from cheap but inaccurate microelectromechanical systems (MEMS) systems that operate by measuring vibrations to

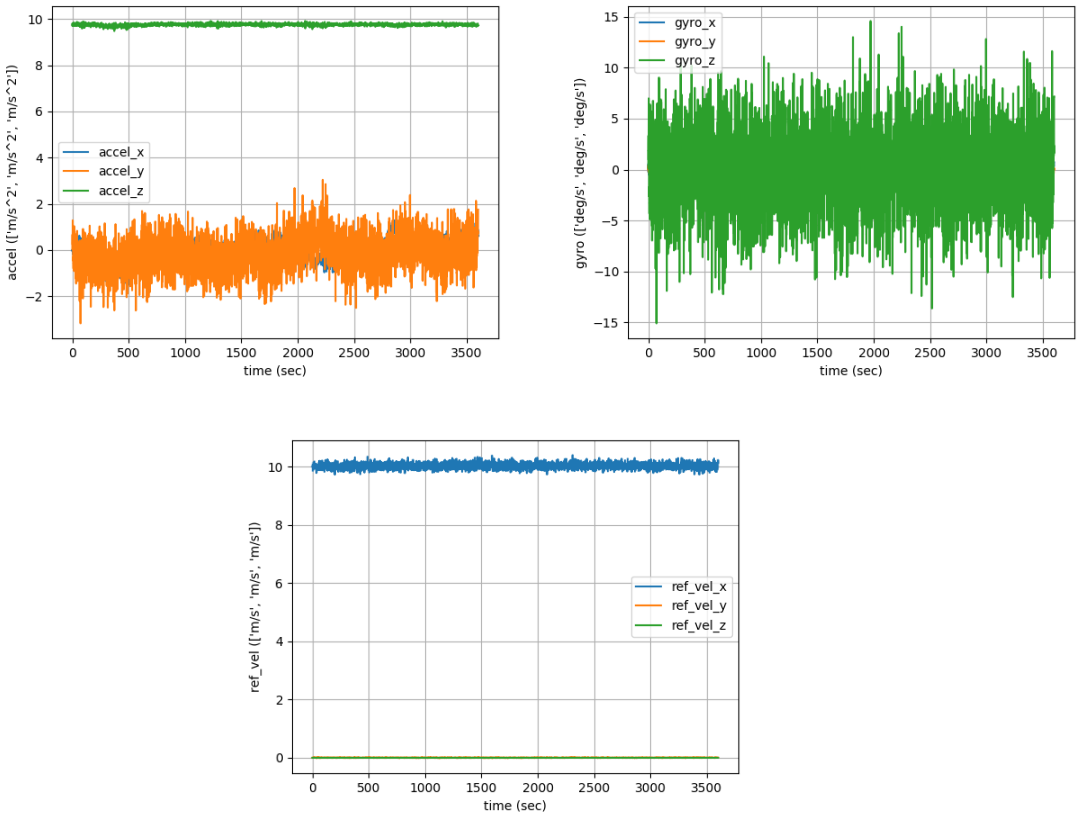

Fig. 1 Graphs of generated accelerometer, gyroscope and velocity data

optical system that operate by measuring the Coriolis effect on a circular light tunnel.[1,4] As pointed out by Gebre-Egziabher,[4] a single error model is unlikely to be equally accurate across all such types of devices, and a simple error model is unlikely to be effective in long-term simulations. In particular, the model we use here does not account for factors like Schuler oscillations, cross-axis sensitivity, calibration errors, and a host of other factors. Many of these are dealt with extensively in Titterton and Weston.[1]

That said, there are a number of types of noise that can be identified by analyzing the behavior of various IMUs, and these can be incorporated into a simple model that can be parameterized for a range of IMUs. In particular, it is well known that various types of noise-generating processes have distinctive power spectra; these in turn are related to Allan variance, which is a quantity that can be computed in the time domain (see, for example Allan,[5] El-Sheimy et al.[6] and Flenniken et al.[7] for a detailed exploration of Allan variance). Once this is done, various noise sources

Table 1 Error sources associated with various slopes in Allan variance graph

Error Source	AV Slope
Wide-band Noise	$-\frac{1}{2}$
Exponentially Correlated Noise	$\frac{1}{2}$
Rate Random Walk	$\frac{1}{2}$
Linear Rate Ramp	+1
Quantization Noise	-1
Sinusoidal Input	-1
Flicker Noise	0

are found to correspond to particular slopes in the Allan variance graph. Table 1 is reproduced from Gebre-Egziabher.[4] It is worth noting that the slopes are not unique, and thus differing noise sources cannot always be distinguished purely on the basis of Allan variance.

The first two of these types of noise will be incorporated into our noise model. The wide-band noise is simply white noise; the exponentially correlated noise is modelled by a first-order Gauss-Markov process, which we now discuss.

2.4.1 Auto-regressive Noise and the First-Order Gauss-Markov Model

One basic property possessed by many time series is being *stationary* – this means the statistical properties of the series remain constant over time.[8] Specifically, a process $\{ z_t \}$ is stationary if, given the joint probability distribution $p(z_{t_1}, z_{t_2})$, we have $p(z_{t_1}, z_{t_2}) = p(z_{t_1+_h}, z_{t_2+h})$ for any times t_1, t_2 and offset h (i.e., the distribution depends only on the temporal difference). A number of important time series are stationary—in particular, of the types of noise processes described in Section 2.4 , wide band noise and exponentially correlated noise both exhibit this property.

One crucial property of stationary processes is that they are determined by their mean, covariance, and *autocorrelation sequence* (defined in the next paragraph).[8] Thus, specifying these quantities for a stationary noise sequence will determine the sequence.

For a fixed integer k, we define the *autocorrelation at lag* k as the correlation between z_t and z_{t+k} (this is constant across all values of t because the process is stationary). Specifically, if μ is the mean of the sequence and σ_z is its standard

deviation, then

$$\begin{aligned}\rho_k &= \frac{E[(z_t-\mu)(z_{t+k}-\mu)]}{\sqrt{E[(z_t-\mu)^2(z_{t+k}-\mu)^2]}} \\ &= \frac{E[(z_t-\mu)(z_{t+k}-\mu)]}{\sqrt{E[(z_t-\mu)^2(z_t-\mu)^2]}} \\ &= \frac{E[(z_t-\mu)(z_{t+k}-\mu)]}{\sigma_z^2}\end{aligned} \tag{1}$$

We note that a white noise sequence $z_k \sim \mathcal{N}(0, \sigma^2)$ is an exemplary stationary process with

$$\rho_k = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}$$

since the z_k are pairwise uncorrelated.

By replacing the discrete sequence $\{ z_k \}$ with continuous functions $z(t)$ and using the continuous analogues of the functions involved, one can similarly define an *autocorrelation function* $R_z(t)$ which intuitively measures the correlation between $z(s)$ and $z(s+t)$*. This allows us to study the autocorrelation function of a first-order Gauss-Markov model, which is a continuous process model determined by the following parameters:

- A time correlation constant τ.
- A variance parameter σ^2 that controls how much noise is present in the process.

Let us define $\beta := \frac{1}{\tau}$. The continuous first-order Gauss-Markov process is defined by the following differential equation.

$$\dot{z}(t) = -\beta z(t) + \sqrt{2\beta\sigma^2} w(t) \tag{2}$$

where $\omega(t)$ is standard white noise — that is, $\omega(t) \sim \mathcal{N}(0, 1)$. The variable β determines the autocorrelation function and it can be shown that $R_z(t) = \sigma^2 e^{-\beta|t|}$. See Fig. 2 for a graph of the autocorrelation of a a first-order Gauss-Markov process

*Since we deal with stationary processes, this autocorrelation is simply a function of t.

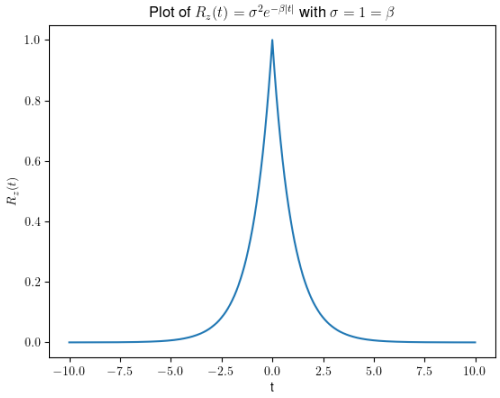

Fig. 2 Autocorrelation of a first-order Gauss-Markov process

We can thus think of β (and hence τ) as determining the rate at which $R_z(t)$ decays: larger values of β indicate a more rapid decorrelation between $z(0)$ and $z(t)$. Thus, larger values of τ mean that the correlation between $z(0)$ and $z(t)$ will decay more slowly as a function of k. In particular, it can be shown that as $\tau \to \infty$ the process becomes closer to white noise. It can also be shown[9] that the variance of a first-order Gauss-Markov process over time interval T is given by

$$\sigma^2 \left(1 + \frac{1 - e^{-2T\beta}}{2T\beta}\right)$$

For a fixed T, this is a decreasing function of β and hence an increasing function of τ. Thus, the process will be noisier for larger values of τ.

If we assume the functions of t on the right-hand side of Eq. 2 are constant from $t = k$ to $t = k + 1$, then, letting ω_k be a sample of $\omega(t)$ at time k, we can discretize as follows

$$\begin{aligned}
\int_{t=k}^{t=k+1} \dot{z}(t)\, dt &= \int_{t=k}^{t=k+1} \left(-\beta z(t) + \sqrt{2\beta\sigma^2}\omega(t)\right)\, dt \\
z_{k+1} - z_k &= \int_{t=k}^{t=k+1} \left(-\beta z_k + \sqrt{2\beta\sigma^2}\omega_k\right)\, dt \\
&= -\beta\Delta t z_k + \sqrt{2\beta\sigma^2}\Delta t\omega_k
\end{aligned}$$

This yields:

$$z_{k+1} = (1 - \beta\Delta t)z_k + \sqrt{2\beta\sigma^2}\Delta t\omega_k \tag{3}$$

where again $\omega_k \sim \mathcal{N}(0, 1)$.

With Eq. 3 we have a clear recurrence relation that we can use to calculate z_k for arbitrary k, using the initial condition $z_0 = 0$.

2.4.2 Final IMU Noise Model

Our final model used the following proposed model of Gebre-Egziabher[4] for general IMU noise:

$$b(t) = b_0 + b_1(t) + b_\omega(t) \tag{4}$$

That is, the bias at time t is determined by the following:

1. b_0 is a constant bias. The constant bias is a bias that might come from a miscalibration or some other source of error. It is easily computable by running the sensor without any movement, and since it is constant it can simply be subtracted from the sensor's input.

   This is safely assumed to be zero since it can be physically measured* and subtracted from the sensor's input.

2. $b_1(t)$ is a first-order Gauss-Markov model (corresponding to the exponentially correlated noise of Table 1†).

   Gebre-Egziabher[4] notes:

   > The $b_1(t)$ term is a time varying bias and can be viewed as having two components. The first component accounts for non-deterministic output errors. The second component accounts for output errors due to external factors such as temperature. It is possible to identify these external factors and develop an accurate model describing their effects on the gyro output. If one compensates for the output errors caused by these external

*Gebre-Egziabher[4]: "For example, it can be obtained by taking long term data when the sensor is subjected to a zero input; the average of the long term data will be b_0. It can also be computed on-line using an estimator that is part of the very navigation system that employs the sensor."

†We note that the term comes from the autocorrelation function having an exponential distribution.

factors, then the only remaining gyro errors would be stochastic. As will be shown shortly, however, it is not easy to compensate for these external factors.

3. $b_\omega(t)$ represents the sampling noise and corresponds to wide-band or band-limited white noise. According to Gebre-Egziabher,[4] this is usually high-frequency relative to the noise in $b_1(t)$, and thus can be measured by sampling the sensor at relatively high rate.

It is worth mentioning that more accurate noise models are obtainable by using the non-stationary autoregressive moving average (ARMA) model, as argued for in Nassar.[10] This comes at the expense of some computational cost, and does not fit the standard model provided by Scorpion. It may, however, be worth exploring in future work.

Properly fitting a noise model to a given sensor requires analyzing a large amount of sensor data, which we do not have access to; therefore, we employ the given parameters and tune the value τ as appropriate.

For specific IMU noise models, we chose specific units to act as exemplars "high", "mid", and "low" accuracy IMUs. For the high-grade unit we chose the Honeywell HG9900[11]—this is considered a "navigation" grade IMU. According to VectorNav Techonologies,[12] "These systems cost around $100,000 and have a size of about 6in x 6in x 6in." For the medium-grade unit we used the Honeywell HG1930[13] and for the low-grade we used the Bosch BMI088[14] — the last was chosen because of its similarity to the IMUs in current smartphones.

Noise information corresponding to the standard deviation in the Gauss-Markov process is available from the data sheets and was used in our modelling. The correlation coefficient τ is not available on the data sheets, and is usually set somewhat arbitrarily in models (e.g., $\tau = 3600$ for Scorpion's model of an HG9900 whereas GNSS-INS-SIM uses $\tau = 100$ for the same IMU). We ran several experiments and found that using $\tau = 100$ gave a total drift after 1 h that corresponded well with the expected drift published in the data sheets.[11,13,14]

Note that in the Tables 2 and 3, σ_1, τ_1 are the parameters for the exponentially correlated noise while σ_ω is the standard deviation for the wide-band noise (all

angles are in degrees).

Table 2 Accelerometer specifications. Parameters are equal in all three components. The σ_1, σ_ω parameters are reported in meters per seconds-squared and meters per square-root-of-seconds-cubed, respectively; τ is reported in seconds.

Grade	Model Name	σ_1	τ_1	σ_ω
High	HG9900	2.45e-4	100.0	0
Mid	HG1930	0.0029	100.0	0.0015
Low	BMI088	0.0196	100.0	0.00225

Table 3 Gyroscope specifications. Parameters are equal in all three components unless otherwise noted. The σ_1, σ_ω parameters are reported in degrees per second and degrees per square-root of seconds, respectively; τ is reported in seconds.

Grade	Model Name	σ_1	τ_1	σ_ω
High	HG9900	9.72e-7	100.0	3.32e-5
Mid	HG1930	0.0002	100.0	[0.00208, 0.0015, 0.0015]
Low	BMI088	0.00056	100.0	0.014

2.5 Visual Odometry Noise

Rather than referring to a specific localization technique, visual odometry refers to an array of techniques in which a temporal sequence of images is used to estimate motion and thus relative position. We adopt a relatively simple model of visual odometry in which, at each timestep, the system receives the ground truth relative poses with Gaussian noise added. That is, if dP represents the relative change in position ($dP \in \mathbf{R}^3$; assuming NED coordinates) and $d\theta$ represents the relative change in attitude ($d\theta \in \mathbf{R}^3$, assuming Euler angle coordinates), then for some $\sigma_P, \sigma_\theta > 0$, the system will receive a pose change represented $dP + \epsilon_P, d\theta + \epsilon_\theta$ where $\epsilon_P \sim \mathcal{N}(0, \sigma_P), \epsilon_\theta \sim \mathcal{N}(0, \sigma_\theta)$. In order to calculate σ_p and σ_θ, we computed summary statistics from the KITTI Odometry Benchmark,[15] published at http:// www.cvlibs.net/datasets/kitti/eval_odometry.php. Specifically, we determined that high-, mid-, and low-grade visual odometry should correspond respectively to errors of $1\%, 2\%$ and 5%. The values for σ_p, σ_θ given in Table 4 were determined to give these results.

We note that while the actual noise associated with a visual odometry algorithm is usually not Gaussian, Gaussian noise represents the best-case scenario for the filtering-based approach we study here. Thus, we would expect that the error associated with visual odometry and Kalman filtering reported in the next section represents a lower bound on the expected error that would occur in the wild.

Table 4 Visual odometry noise parameters

Grade	σ_P	σ_θ
High	0.005	0.0005
Mid	0.01	0.001
Low	0.1	0.005

It is also worth noting that according to Liu et al.,[16] relative pose error *can* be modelled by a Gaussian. The following justifications are given by Liu et al.:

- The empirical study in Williams and Reid[17] demonstrates a Gaussian noise model.
- Many of the noise input disturbances (e.g., feature localization uncertainty, calibration uncertainty) are approximately Gaussian and since Gaussians are preserved under linear transformations, we can expect the total error to be largely Gaussian.
- For large-scale visual odometry, the number of inputs is very large and hence the central limit theorem guarantees that the average noise will be Gaussian.

2.6 Clock Noise

As with IMU, there are a number of possible models for clock noise. One option is to use a first-order Gauss-Markov model, another is simply to use white noise. In this preliminary study, we introduced clock noise as a Gaussian $\mathcal{N}(0, 0.0001)$—that is, at every timestep when dt seconds had elapsed in ground-truth time, the system received a relative time change of $dt + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, 0.0001)$.

2.7 Estimated Trajectory Generation with Simulated IMU Reckoning

Starting with a given initial position and velocity p_0, v_0 along with simulated IMU data $\mathbf{A_N}, \mathbf{G_N}$, we compute a corresponding trajectory by the process of *mechanization*.[1] This is essentially a process of performing a double integration on the acceleration data to estimate a change in position and an integration of the gyroscope data to estimate a change in attitude; this is complicated by a number of factors including accounting for skewed readings due to gravity and the Earth's rotation, along several coordinate transformations necessitated by the inertial units operating in a fixed inertial coordinate system while the end results should be relative to the Earth.

2.8 Estimated Trajectory Generation with Simulated Visual Odometry

As discussed previously, visual odometry uses camera information to estimate the relative pose change (changes in position and attitude) between successive frames. As the techniques are varied, we give a generic model that produces the true relative pose change (in radians for the attitude change and meters for the position change) with white noise added.

2.9 Estimated Trajectory Generation with Kalman Filtering

We have until now enumerated two distinct ways for a system to estimate its position: by mechanizing inertial measurements and by using visual odometry. These techniques are combined in a method known as *visual inertial odometry* (VIO). VIO algorithms can be further classified as tightly or loosely coupled. The distinction is that while the former class of algorithms combines visual and inertial information early into a joint representation in which the two are usually indistinguishable, loosely coupled algorithms represent the information from each sensor separately. Because the contributions from various components are easier to parse out, this report will focus on loosely coupled algorithms. In particular, we will employ an *extended Kalman filtering* algorithm that computes a weighted average of the results of the visual odometry with the results of mechanization. The details are as follows, and are largely rephrased from Titterton and Weston[1] and Harvey.[18]

Our main task is to keep track of the position of the navigation system at time t. This will clearly involve keeping track of the system's coordinates; since we want to use a Markov process to make our computations tractable, we will also have to keep track of all the variables that will affect the change in position. We use the "Pinson15" state model given in Titterton and Weston.[1] In particular, we will use a state model that separates into two components: $\vec{x}_t$ will represent the current state of the system and $\vec{u}_t$ will represent the input to the system.

Thus we define

$$\begin{aligned} \vec{x}_t &:= \begin{bmatrix} \alpha & \beta & \gamma & v_N & v_E & v_D & L & l & h \end{bmatrix}^T \\ \vec{u}_t &:= \begin{bmatrix} \omega_x & \omega_y & \omega_z & f_x & f_y & f_z \end{bmatrix}^T \end{aligned} \tag{5}$$

where

- (α, β, γ) represent the attitude of the system in terms of roll, pitch, and yaw.
- (v_N, v_E, v_D) represent the velocity of the system in the NED coordinate system.
- (L, l, h) represent the position in the latitude-longitude-altitude coordinate system.
- $(\omega_x, \omega_y, \omega_z)$ represent the rotational force applied to the system at time t.
- (f_x, f_y, f_z) represent the specific force ($\mathbf{f} = \mathbf{a} - \mathbf{g}$) applied to the system at time t.

Ideally, this should be enough information to determine the state update $\vec{x}_{t+1}$. In a navigational context, the effects of the $\vec{x}_t$ and $\vec{u}_t$ on the system are representable as vectors, so we can assume that their cumulative effect is represented by

$$\vec{x}_{t+1} = \mathbf{f}(\vec{x}_t) + \mathbf{g}(\vec{u}_t) \tag{6}$$

for some deterministic vector-valued functions $\mathbf{f}, \mathbf{g}$. The functions $\mathbf{f}$ and $\mathbf{g}$, as discussed below, are specified by instantiating physical laws. Thus, in a world of perfect information, knowing $\vec{x}_t, \vec{u}_t$ would allow us to compute $\vec{x}_{t+1}$.

Due to the complexity of the real world, we usually do not know what $\vec{x}_t$ $\vec{u}_t$ are exactly, but must rather work with our best estimates $\hat{\vec{x}}_t, \hat{\vec{u}}_t$. In the context of Kalman filtering, these are assumed to differ from the truth by some specified white noise:

$$\begin{aligned} \hat{\vec{x}}_t &= \vec{x}_t + \epsilon_t \\ \hat{\vec{y}}_t &= \vec{y}_t + \eta_t \end{aligned} \tag{7}$$

where for some diagonal covariance matrices Σ_x, Σ_u, we have $\epsilon_t \sim \mathcal{N}(0, \Sigma_x), \eta_t \sim \mathcal{N}(0, \Sigma_y)$.

Suppose that, periodically, the system also receives noisy measurements of the state: $\hat{\vec{y}}_t = \mathbf{h}(\vec{x}_t) + \nu_t$. If the function $\mathbf{h}$ has a computable inverse, then we can estimate $\vec{x}_t$ by computing $\mathbf{h}^{-1}(\hat{\vec{y}}_t)$. If $\mathbf{h}$ also happens to be linear, then $\mathbf{h}^{-1}$ will be as well and we will have

$$\mathbf{h}^{-1}(\hat{\vec{y}}_t) = \vec{x}_t + \mathbf{h}^{-1}(\nu_t)$$

that is, the inverse image of $\hat{\vec{y}}_t$ will differ from $\vec{x}_t$ by some Gaussian noise. As such, we can get an estimate of $\vec{x}_t$ by averaging $\mathbf{h}^{-1}(\hat{\vec{y}}_t)$ with $\hat{\vec{x}}_t$ and the resulting covariance on the estimate will be lower than the covariance on either $\mathbf{h}^{-1}(\hat{\vec{y}}_t)$ or $\hat{\vec{x}}_t$. The idea of Kalman filtering is to extend this observation, and assuming that $\mathbf{f}$ and $\mathbf{h}$ are linear (and $\mathbf{h}$ is not necessarily invertible), find an optimal weighted sum of the estimates provided by $\mathbf{f}(\hat{\vec{x}}_t)$ and $\hat{\vec{y}}_{t+1}$ to determine $\hat{\vec{x}}_{t+1}$.

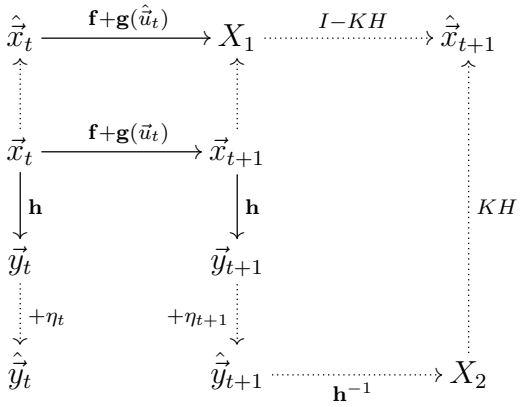

Fig. 3 Kalman filtering

Let us, then, suppose that $\mathbf{f}, \mathbf{g}, \mathbf{h}$ are linear functions with respective matrix representations F, G, H. Let $X_1 = F\hat{\vec{x}}_t + Gu_t$ and let X_2 be any vector such that $HX_2 = \hat{\vec{y}}_{t+1}$ (if $\mathbf{H}$ is invertible, then we will have $X_2 = H^{-1}(\hat{\vec{y}}_{t+1})$). Let $\vec{z}_{t+1} = H\hat{\vec{x}}_{t+1} - \hat{\vec{y}}_{t+1}$; this will be an estimate of the error in $\hat{\vec{x}}_{t+1}$. Then $\vec{z}_{t+1} = H(X_1 - X_2)$, and we want to find some weight matrix K so that the variance of

$$\hat{x} = X_1 - KH(X_1 - X_2) \tag{8}$$

is minimized (in particular, the trace of the covariance matrix of $\hat{x}$ is minimized[19]). The matrix K is called the matrix of *Kalman gains*. If the covariance matrices of X_1 and y_{t+1} are respectively denoted by P, R then it can be shown (e.g., Harvey,[18] Reid[19]) that variance of $\hat{x}$ is minimized by choosing

$$\begin{aligned} P' &= FPF^T + \Sigma_y \\ P_{new} &= P' - KHP' \end{aligned} \tag{9}$$

and updating K by

$$K = P'H^T[HP'H^T + R]^{-1} \tag{10}$$

and $\hat{x}$ by

$$\hat{\vec{x}}_{t+1} = X_1 - K\vec{z}_{t+1} = F\hat{\vec{x}}_t - K[HF\hat{\vec{x}}_t - \vec{y}_{t+1}] \tag{11}$$

This process is summarized in Fig. 3.

It is worth noting that, in addition to minimizing the variance, it is provable that under the assumptions given, the Kalman filter produces a maximum likelihood estimate for the state (conditioned on X_1, X_2).[18]

As will be discussed more fully in the next section, in the navigational context the function $\mathbf{f}$ is *not* linear. Since it is smooth, one uses the linear approximation derived from the Jacobian at time t

$$F_t = \left.\frac{d\mathbf{f}}{dt}\right|_{\hat{\vec{x}}_t} \tag{12}$$

and similarly for G_t ($\mathbf{h}$ *is* linear in our situation and need not be approximated). This modification is known as *extended Kalman filtering* in the literature.

It is important to note that the non-linearity means that optimality is not preserved, and indeed in the context of navigation it is understood that the extended Kalman filter does not even provide an unbiased estimate (see Bailey et al.[20] and Castellanos et al.[21]).

In our particular case, we will want $\mathbf{f}$ to represent the *navigation equations*, which specify how all 15 components of our state change from time t to time $t + 1$ based on the information in $\vec{x}_t$, while $\mathbf{g}$ represents the impact of extra forces applied to the system (usually this will consist of the specific force along with rotational changes). The calculations involved in specifying these functions properly are rather complex, occupying the greater part of Titterton and Weston,[1] a 500-page reference on the topic.

We will use the following for F and H. F summarizes the often complex physics of navigation; it is given by

$$\begin{pmatrix}
0 & -A & \frac{v_N}{R} & 0 & \frac{1}{R} & 0 & -\Omega \sin L & 0 & \frac{v_E}{R^2} \\
A & 0 & B & -\frac{1}{R} & 0 & 0 & 0 & 0 & \frac{v_N}{R^2} \\
-\frac{v_N}{R} & -B & 0 & 0 & -\frac{\tan L}{R} & 0 & -\Omega \cos L - \frac{v_E}{R \cos^2 L} & 0 & \frac{v_E \tan L}{R^2} \\
0 & -f_D & f_E & \frac{v_D}{R} & -2A & \frac{v_N}{R} & -v_E \left(2\Omega \cos L + \frac{v_E}{R \cos^2 L}\right) & 0 & -\frac{v_E^2 \tan L - v_N v_D}{R^2} \\
f_D & 0 & -f_N & E & \frac{C}{R} & 0 & D & 0 & 0 \\
-f_E & f_N & 0 & \frac{2v_N}{R} & -2B & 0 & 2\Omega v_E \sin L & 0 & \frac{1}{R^2}\left(v_N^2 + v_E^2\right) \\
0 & 0 & 0 & \frac{1}{R} & 0 & 0 & 0 & 0 & -\frac{v_N}{R^2} \\
0 & 0 & 0 & 0 & \frac{1}{R \cos L} & 0 & \frac{v_E \tan L}{R \cos L} & 0 & -\frac{v_E}{R^2 \cos L} \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0
\end{pmatrix}$$

where

- Ω denotes the (latitude-dependent) rate of rotation of the Earth, in radians.
- L is the current latitude of the system
- (v_N, v_E, v_D) give the current coordinates of the system in the NED frame.
- R is an unused term used to model the Schuler oscillation; we set $R := 1$.
- $A = (\Omega \sin L + \frac{v_E}{R} \tan L)$
- $B = \Omega \cos L + \frac{v_E}{R}$
- $C = (v_N \tan L + v_D)$
- $D = (2\Omega(v_N \cos L - v_D \sin L) + \frac{v_N v_E}{R \cos^2 L})$
- $E = (2\Omega \sin L + \frac{v_E}{R} \tan L)$

The Scorpion software package[22] has implemented most of the necessary computations, obviating the need for detailed interaction with them. We emphasize, however, that a casual observation reveals that the function $\mathbf{f}$ is far from linear, and we

can therefore expect fairly serious bias to be introduced by linearizing. In particular, the optimality of the Kalman filter, which relies on the linearity of the function $\mathbf{f}$, is no longer guaranteed.

Finally, the matrix H should be such that, given the true state $\vec{x}_t$ we would have $\vec{y}_t = H\vec{x}_t$, where in our situation $\vec{y}_t$ represents the true visual odometry measurement. Recall that visual odometry algorithms take as input at time $t+1$ the *relative positions* $p_{t+1} - p_t$ and *relative attitudes* $\alpha_{t+1} - \alpha_t$. For somewhat technical reasons internal to Scorpion* we can simply take H to be the identity matrix.

3. Detailed Results

We give and interpret the results of our simulation. We note that while it is common in empirical settings to "tune" a Kalman filter by estimating the initial covariance matrices P_0, R_0 for the state and measurement noise, this is a priori not necessary in our situation since the noise parameters are known. However, since we are working in the non-linear situation of an extended Kalman filter, it is possible that some tuning of the covariances may help mitigate the error introduced by the non-linearity of $\mathbf{f}$.

We begin by noting Fig. 4, which displays various trajectories in two and three dimensions. The trajectories present in these graphs are described as follows.

- *Truth.* This is the ground truth trajectory.
- *VO.* This is the trajectory determined by the visual odometry.
- *IMU Mech.* This is the trajectory determined by mechanizing the (noisy) IMU inputs.
- *Filter.* This is the trajectory determined by filtering the IMU mechanizations against the visual odometry measurements.
- *Truth abs.* This is the trajectory of the ground truth computed by summing relative poses.
- *Truth raw imu.* This is the trajectory computed by mechanizing ground truth accelerations and attitude changes.

*In particular, because Scorpion filtering uses $\delta\hat{\vec{x}}_t$; the estimated *error* in $\hat{\vec{x}}_t$ at time t. This quantity is the same whether we are using absolute poses or relative pose changes.

Both show that the filter ends up having approximately the same shape as the ground truth and that overall the visual odometry corresponds much more closely to the truth than the result of pure IMU mechanization (this is unsurprising, given the relatively low noise variance on the visual odometry).

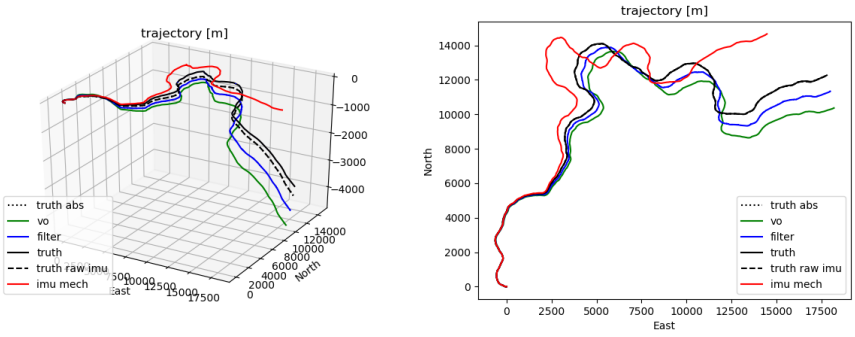

Fig. 4 Graphs of trajectory in three and two dimensions, computed by simulating an HG9900 IMU

We also note that the trajectories plotted with dotted lines are not visible in the graphs—this is desirable because they represent potential deviations from expectation.

Table 5 presents the overall errors (along with their standards deviations) from running pure IMU mechanization, pure visual odometry, and a filtered average of these two averaged over 30 trials.

We note first that the standard deviations associated with process are very large. This is at some level unsurprising since the integration of raw errors corresponds to a kind of random walk, which is known to have a variance that increases linearly with the number of steps.

We also note that for high-grade IMUs, the filters outperform both as expected. This effect degrades with the quality of the IMU. For example, the low-grade IMU with low-grade visual odometry driving simulation produces a filter error that is almost 1.5 times as high as the visual odometry error. As it turns out, this result accords with the literature on using extended Kalman filters for navigation with low-grade IMUs. In particular, Bailey et al.,[20] Castellanos et al.,[21] and Julier and Uhlmann[23]

all cite total heading error combined with linearization error as particular causes of filter inconsistency in a navigational framework. To test this, we reran the low-grade filter simulations using a straight trajectory; this has the effect of minimizing linearization error since the trajectory is in fact linear. We obtain a visual odometry error of 3331.8 m, an IMU mechanization error of 24078339.0 m and a filter error of 3255.9 m; thus, without linearization error, the filtering does seem to outperform both independent sources as expected. We take this as evidence that the errors observed are indeed a result of gyroscopic noise combined with linearization error.

Finally, we note that even in our best-case scenario of a pedestrian using a high-grade IMU with high-quality visual odometry, we can still expect 164 m of filtered error after 1 h. Thus filtering by itself seems unable to meet our desired accuracy.

Table 5 Summary simulation results

IMU	VO	VO Error (Std)	IMU Error (Std)	Filter Error (Std)
High	High	256.86 (247.7)	8486.63 (4235.5)	164.49 (86.2)
High	Mid	584.08 (281.7)	8212.52 (3839.6)	295.70 (112.8)
High	Low	2352.69 (1664.4)	9776.56 (4945.3)	912.84 (590.2)
Mid	High	347.79 (284.8)	1032601.78 (393455.6)	118.14 (58.5)
Mid	Mid	601.54 (345.1)	1022745.22 (504039.3)	126.82 (84.8)
Mid	Low	2713.38 (1428.7)	1133439.73 (508452.2)	1013.71 (491.4)
Low	High	338.06 (197.7)	27083455.18 (14512700.5)	3921.68 (2188.4)
Low	Mid	555.05 (382.5)	29172981.29 (14328055.9)	3767.04 (1404.7)
Low	Low	2474.22 (1157.4)	25943344.88 (13308989.7)	9616.50 (6347.4)

(a) Summary simulation results – Walking

IMU Grade	VO Grade	VO Error (Std)	IMU Error (Std)	Filter Error (Std)
High	High	1386.0 (944.1)	8605.8 (3591.9)	460.4 (237.3)
High	Mid	2546.8 (1522.3)	9179.8 (3795.6)	839.3 (406.6)
High	Low	10808.4 (5401.8)	7757.6 (3653.0)	3306.3 (1542.4)
Mid	High	1177.7 (715.5)	1049232.9 (539317.5)	699.1 (418.3)
Mid	Mid	2629.8 (1381.6)	948016.5 (457325.6)	676.1 (510.1)
Mid	Low	11522.3 (6853.9)	1036500.2 (578554.2)	1941.1 (975.6)
Low	High	1216.1 (845.2)	27302116.2 (14350836.9)	17995.9 (10008.3)
Low	Mid	2742.5 (1409.2)	24325928.3 (10488808.8)	15672.6 (6817.3)
Low	Low	11208.7 (5649.9)	27529088.8 (10760715.5)	18724.1 (9261.9)

(b) Summary simulation results – Driving

3.1 Other Statistics

We examine two other statistics that measure navigational accuracy. The Circular Error Probable is reported in Table 6, and the number of seconds until the error is

past 10 m is reported in Table 7. The former quantity gives the radius of a circle (in our case a sphere) centered at the ground truth that would statistically be expected to contain half of all trajectories—it corresponds to the median of the errors over 30 trials.

As in Tables 5a and 5b, the results show that the filtering studies will attain an unacceptable error level very quickly and can be expected to be fairly far off the truth in the course of an hour.

Table 6 Circular error probable

IMU	VO	VO CEP	IMU CEP	Filter CEP
High	High	223.4	8377.9	148.3
High	Mid	572.3	7641.447376	268.643095
High	Low	1987.6	7577.2	792.55639
Mid	High	242.6	1096939.0	106.8
Mid	Mid	546.8	943597.9	108.7
Mid	Low	2264.6	1087036.0	1009.4
Low	High	334.4	23405700.0	3648.4
Low	Mid	454.1	26665870.0	3473.7
Low	Low	2386.7	25365570.0	8891.1

(a) Circular error probable results – Walking

IMU Grade	VO Grade	VO CEP	IMU CEP	Filter CEP
High	High	1214.1	8343.6	431.5
High	Mid	1771.1	8287.2	788.4
High	Low	9752.8	7618.5	3517.2
Mid	High	990.5	900188.6	599.8
Mid	Mid	2629.8	894692.5	587.4
Mid	Low	9798.8	878693.2	1708.5
Low	High	928.1	22693700.0	17995.9
Low	Mid	2389.6	22709070.0	13184.6
Low	Low	10797.0	24474800.0	14496.9

(b) Circular error probable results – Driving

3.2 Clock Noise

Preliminary simulations with the clock errors make it clear that this is a very significant source of error. For a single trial with high-grade IMUs and visual odometry, a driving simulation produced 614.5 m of visual odometry error, 24230352.9 m of IMU mechanization error, and 3649.3 m of filter error. This is with a relatively small amount of clock noise variance of 0.0001 seconds per step for every 0.01-s update, or 0.01 per second.

Table 7 Time to 10-m error (in seconds)

IMU	VO	VO T10	IMU T10	Filter T10
High	High	316.5	216.7	577.3
High	Mid	175.7	255.8	297.3
High	Low	60.0	259.0	144.65
Mid	High	353.7	58.1	398.8
Mid	Mid	220.4	55.7	258.2
Mid	Low	61.1	56.1	58.3
Low	High	368.4	19.1	76.4
Low	Mid	229.0	18.7	51.7
Low	Low	53.5	18.4	19.9

(a) Average time to 10-m error – Walking

IMU Grade	VO Grade	VO T10	IMU T10	Filter T10
High	High	127.3	254.8	265.0
High	Mid	72.39	265.9	197.5
High	Low	24.4	275.8	124.4
Mid	High	113.7	57.2	226.7
Mid	Mid	69.2	57.6	174.8
Mid	Low	23.4	58.8	45.7
Low	High	139.8	18.6	41.2
Low	Mid	73.0	19.2	39.3
Low	Low	23.4	18.9	18.0

(b) Average time to 10-m error – Driving

Figure 5 shows a graph of the trajectories (left) and the ground truth with the filtered result (right). One can see that the clock noise causes the IMU estimates to veer enough off course that the ground truth graph is actually lost in the scale, while the second graph shows that the filter is able to maintain the proper scale but loses a great deal of accurate heading information.

4. Conclusion

Overall, we note that errors accumulate rapidly in even the best-case scenario and that these are only partially ameliorated by filtering. It is also striking that overall visual odometry seems to accumulate errors much less rapidly than inertial integration, even when the former has significantly higher covariance. We suspect that this is due to the relative simplicity of the visual odometry paradigm. One need only add estimated pose changes and the algorithm is relatively immune to the fluctua-

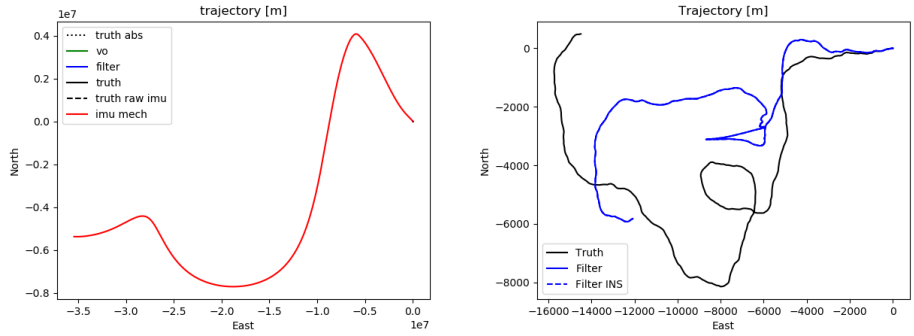

Fig. 5 Graphs of trajectory with clock noise showing IMU mechanization (left) and just the ground truth and filtered result (right)

tions that can come from factors such as the miscalculation of gravity or the Earth's rotation.

In summary, the main conclusions from this study are as follows:

- By itself, filtering is not capable of localizing a system with the necessary accuracy.
- The error accumulated by visual odometry is significantly lower than that accumulated by raw inertial measurements.
- With current technology, we can expect to have a 10-m drift within 10 min, even in the best-case scenario.
- The most promising avenue to close the current gap seems to be to use opportunistic information to mitigate the long-term effects of error drift.

5. References

1. Titterton D, Weston J. Strapdown inertial navigation technology. 2004.

2. ACEINNA. GNSS-INS-SIM: Open-source GNSS + inertial navigation, sensor fusion simulator. Motion trajectory generator, sensor models, and navigation. ACEINNA. 2018 [accessed 2021]. http://github.com/Aceinna/gnss-ins-sim/.

3. O'Dwyer A. Handbook of PI and PID controller tuning rules. Imperial College Press; 2009.

4. Gebre-Egziabher D. Design and performance analysis of a low-cost aided dead reckoning navigator [thesis]. Stanford University; 2003.

5. Allan DW. Statistics of atomic frequency standards. Proceedings of the IEEE. 1966;54(2):221–230.

6. El-Sheimy N, Hou H, Niu X. Analysis and modeling of inertial sensors using allan variance. IEEE Transactions on Instrumentation and Measurement. 2007;57(1):140–149.

7. Flenniken W, Wall J, Bevly D. Characterization of various IMU error sources and the effect on navigation performance. In: Proceedings of the 18th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2005); 2005 Sep; Long Beach, CA. p. 967–978.

8. Box GE, Jenkins GM, Reinsel GC, Ljung GM. Time series analysis: forecasting and control. John Wiley & Sons; 2015.

9. Herring T. Principles of the global positioning system: Lecture 13. Massachusetts Institute of Technology. 2014 [accessed: 2019 Aug 14]. http://www-gpsg.mit.edu/%7Etah/12.540/12.540.Lec13.pdf.

10. Nassar S, Schwarz KP, El-Sheimy N, Noureldin A. Modeling inertial sensor errors using autoregressive (AR) models. Navigation. 2004;51(4):259–268.

11. HG9900 Inertial Measurement Unit. Honeywell Aerospace. [accessed: 2019-08-12]. https://aerospace.honeywell.com/en/%7E/media/aerospace/files/brochures/n61-1638-000-000-hg9900inertialmeasurementunit-bro.pdf.

12. Inertial Measurement Units and Inertial Navigation. VectorNav Technologies. [accessed: 2019 Aug 12]. https://www.vectornav.com/support/library/imu-and-ins.

13. HG1930 Inertial Measurement Unit. Honeywell Aerospace. [accessed: 2019 Aug 12] https://aerospace.honeywell.com/en/%7E/media/aerospace/files/brochures/n61-1637-000-000-hg1930inertialmeasurementunit-bro.pdf.

14. BMI088 Inertial Measurement Units. Bosch Sensortec. [accessed: 2019 Aug 12]. https://www.bosch-sensortec.com/bst/products/all_products/bmi088_1.

15. Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR); 2012 June; Providence. p. 3354–3361.

16. Liu H, Jiang R, Hu W, Wang S. Navigational drift analysis for visual odometry. Computing and Informatics. 2014;33(3):685–706.

17. Williams B, Reid I. On combining visual slam and visual odometry. In: 2010 IEEE International Conference on Robotics and Automation; 2010 May; Anchorage. p. 3494–3500.

18. Harvey AC. Forecasting, structural time series models and the Kalman filter. Cambridge University Press; 1990.

19. Reid I. Estimation II. Oxord University. [accessed: 2019 Aug 13]. http://www.robots.ox.ac.uk/%7Eian/Teaching/Estimation/LectureNotes2.pdf.

20. Bailey T, Nieto J, Guivant J, Stevens M, Nebot E. Consistency of the EKF-SLAM algorithm. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2006 October; Beijing. p. 3562–3568.

21. Castellanos JA, Neira J, Tardós JD. Limits to the consistency of EKF-based SLAM. IFAC Proceedings Volumes. 2004;37(8):716–721.

22. Scorpion: a software framework for Bayesian navigation. [accessed 2021]. Air Force Institute of Technology.

23. Julier SJ, Uhlmann JK. A counter example to the theory of simultaneous localization and map building. In: Proceedings 2001 ICRA. IEEE International

Conference on Robotics and Automation (Cat. No. 01CH37164); Vol. 4; 2001 May; Seoul. p. 4238–4243.

List of Symbols, Abbreviations, and Acronyms

TERMS:

GPS	global positioning system
IMU	inertial measurement unit
LLA	latitude-longitude-altitude coordinate system
NED	north-east-down
MEMS	microelectromechanical systems
PID	proportional-integral-derivative
VIO	visual inertial odometry

MATHEMATICAL SYMBOLS:

$\mathbf{R}$	the set of real numbers
$\mathbf{Z}$	the set of integers
$M(a, b)$	the ring of all $a \times b$ real-valued matrices
$\mathcal{N}(\mu, \sigma^2)$	the Gaussian distribution with mean μ and standard deviation σ
$X \sim D$	X is a random variable which is drawn from distribution D
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	a multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$b(t)$	bias at time t; the noise introduced following the system defined noise model

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB
5 (PDF)	DEVCOM ARL FCDD RLA PB D DONAVANIK J CONROY N SROUR FCDD RLA P W BENARD FCDD RLA CQ A SCHOFIELD