American Journal of Sensor Technology, 2018, Vol. 5, No. 1, 1-6
Available online at http://pubs.sciepub.com/ajst/5/1/1

DOI:10.12691/ajst-5-1-1

Angular Position Estimation of an Inverted Pendulum Using Low-Cost IMUs

Daisuke Aoyagi[1,*], Sukgi Choi[2]

[1]Department of Mechanical and Mechatronic Engineering and Sustainable Manufacturing, California State University, Chico, CA 95929, USA
[2]Department of Electrical and Computer Engineering, California State University, Chico, CA 95929, USA
*Corresponding author: daoyagi@csuchico.edu

Received June 16, 2018; Revised August 21, 2018; Accepted September 09, 2018

Abstract Seeking an affordable solution to measure a bicycle's roll angle, we came across an Inertial Measurement Unit (IMU) BNO055 by Bosch, which contains 3-axis accelerometer, gyroscope, and magnetometer, and is advertised to produce "absolute orientation" by a built-in proprietary "Fusion" algorithm. We found another low-cost IMU, an MPU-9250 from InvenSense, which could also calculate absolute orientation via embedded Fusion software. Being unable to find information about dynamic characteristics of these IMUs in their datasheets, we sought to evaluate them under dynamic conditions, specifically in the estimation of roll angle. We constructed an inverted pendulum as a model of a bicycle, mounted both IMUs on it, and attached a potentiometer to measure actual angular position for reference. Additionally, as an alternative to the proprietary Fusion algorithms, we devised and implemented an Extended Kalman Filter, which, we hypothesized, would perform better than the proprietary Fusion algorithms, because our algorithm incorporated the kinematics of the inverted pendulum while the Fusion algorithm of the IMUs did not. In a series of experiments, we observed a significant time lag, about 0.05-0.1 second, in BNO055's raw acceleration and gyro signals. The BNO055's Fusion responded with similar lag and an offset of 0.5-3°; we also noticed rather unpredictable fluctuation in the output signals, possibly due to its "automatic calibration" feature, which cannot be disabled. The MPU-9250 exhibited better performance than the BNO055 in terms of raw acceleration signals and, particularly, gyro signals. The MPU-9250's Fusion performed somewhat better than BNO055's, typically showing lag of 0.03-0.06 sec and static offset of 0.5-1°. Our implementation of Kalman Filter based on MPU-9250 raw signal performed better than either Fusion algorithm, with about 0.02-0.03 second lag and 0.5-1° offset, supporting our hypothesis. Our next step is to experiment on an actual bicycle in motion.

Keywords: *inertial measurement unit, angular estimation, inverted pendulum, Extended Kalman Filter*

Cite This Article: Daisuke Aoyagi, and Sukgi Choi, "Angular Position Estimation of an Inverted Pendulum Using Low-Cost IMUs." *American Journal of Sensor Technology*, vol. 5, no. 1 (2018): 1-6. doi: 10.12691/ajst-5-1-1.

1. Introduction

We are working on a project that aims to develop a self-balancing riderless bicycle, not unlike the work of Cerone et al. [1]. One of initial steps in the project is to measure the roll angle of the bicycle, which can then be used for feedback control of various forms to stabilize the bicycle.

Noting recent proliferation of low-cost quadcopters, which typically rely on a Micro-Electro-Mechanical Systems (MEMS)-based Inertial Measurement Unit (IMU) to measure and stabilize its orientation in 3D space, we sought to find a low-cost IMU that estimated roll angle sufficiently accurately and was ready to be plugged into the rest of the system.

We came across an IMU, BNO055 from Bosch Sensortec GmbH (Germany) [2]. This IMU was advertised to produce "absolute orientation", including the roll angle, by a proprietary "fusion" algorithm built into the sensor device, which contains 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer (thus called "9-axis" IMU). However, after a quick initial testing, we noticed a considerable time lag in the sensor's output. Looking for the IMU's specifications, we looked into the IMU's datasheet only to find some numbers related to static response. We could not find specifications or information in terms of dynamic response of the device.

We had another MEMS-based 9-axis IMU on hand: an MPU-9250 from InvenSense, Inc. (San Jose, California, USA). In our initial quick testing, this sensor appeared to produce less time lag than the BNO055 did, at least in its raw acceleration and gyro signals. We also learned that this sensor was capable of running embedded Digital Motion Processor (DMP) that could calculate absolute orientation from 6-axis (3-axis acceleration and 3-axis gyro) raw signals and output it in the quaternions, from which the roll angle could be calculated. To access the

DMP feature, we signed up with "InvenSense Developers Corner" and obtained MotionDriver 6.12 Library [3].

We suspect the proprietary Fusion algorithms built into these IMUs may be based on some form of Kalman Filter [4] or other forms of state estimator, but we do not have access to the specific algorithms. Nevertheless, these IMUs are attractive options for our self-balancing bicycle project, mainly due to their low-cost and availability, as long as the resulting angular estimation is sufficiently accurate for the application.

With the application of bicycle stabilization in mind, we studied the dynamic response of the IMUs. We conducted experiments on a simple inverted pendulum as a model of a bicycle. A somewhat similar test apparatus had been used by Leavitt et al. [5], who estimated the angular position of an inverted pendulum by combining slow-acting tilt sensor and a quick responding gyroscope and accelerometer, while a rotary encoder served as a reference for verification purpose. In our setup, we used a potentiometer (rotary variable resistor) as a reference device.

Our objective was to evaluate the performance of BNO055 and MPU-9250, specifically in the estimation of the roll angle. For further comparison, we devised an implementation of Extended Kalman Filter based on the raw signals (accelerations and angular rate) of the IMUs. We hypothesized that our Kalman Filter implementation would produce better performance than the proprietary Fusion algorithms, because our algorithm incorporated the kinematics of the inverted pendulum while the Fusion algorithm of the IMUs did not.

In the following sections, we describe the experimental methods, and present the results with sample data plots.

2. Materials and Methods

2.1. Experimental Setup

The InvenSense/TDK MPU-9250 is available as a single chip for surface mounting from major distributors, such as Digikey.com. For convenience of wiring and prototyping, however, we opted to use a pre-assembled breakout board (~$15) from SparkFun Electronics [6].

With the aforementioned DMP MotionDriver library available via InvenSense website, we configured the MPU-9250 to generate a pulse signal when DMP data became ready. This digital signal was monitored by an Arduino Uno clone (Arduino #1 in Figure 1), which was programmed to read the available data (raw acceleration and gyro signals plus roll angle calculated from the quaternion output of DMP) via I^2C and send it to a Windows 10 PC via USB/Serial connection.

Using the DMP software, we set a 100 Hz sampling rate. Although the DMP Fusion algorithm's highest possible sampling rate was 200 Hz, we could not reliably reach beyond 100 Hz mainly due to the limited speed of communication between the Arduinos and the PC.

For the BOSCH BNO055, again we opted to use a breakout board (~$35) by Adafruit [7], using the driver software also by Adafruit [8], to access the IMU's Fusion signal as well as the raw data. According to BNO055 datasheet, the IMU has an auto-calibration feature that runs in the background to remove offsets for all three sensors: accelerometer, gyroscope, and magnetometer. While this feature cannot be disabled, the datasheet indicated that "some preliminary steps had to be ensured for this automatic calibration to take place". The specific calibration procedures are described below in the Experimental Protocol section.

The BNO055 board was connected via I^2C to another Arduino Uno clone (Arduino #2 in Figure 1), which received the same data-ready signal from the MPU-9250. Triggered and synchronized by the data-ready signal, Arduino #2 read the auto-calibration status and fusion output, as well as the raw acceleration and gyro signals from the BNO055. The data was sent to the PC via USB/Serial.

We mounted both IMU breakout boards on a breadboard, which was in turn mounted on an inverted pendulum, made of a 30x30mm aluminum extrusion and supported by bearings at the bottom with a potentiometer attached to the axis of rotation, as shown in Figure 1.

The potentiometer, configured as a voltage divider, provided reference angular position. Both the voltage divider output and the supply DC voltage (~5V) were measured via analog inputs on a National Instruments (NI; Austin, Texas, USA) USB-6009 DAQ device. The ratio of the two voltages was statically calibrated against known angular positions measured by a Digi-Pas DWL-80E tilt meter for the range of ±30° with a 3rd-degree polynomial curve-fit. Estimated overall uncertainty of the potentiometer angle was ±0.5°.

We wrote a NI LabVIEW program on the PC to read the incoming data from IMUs via USB/Serial, visualize it on screen, and export it to a spreadsheet. The LabVIEW program also acquired the potentiometer voltages from the USB-6009 via USB. Estimated timing error among IMU and the potentiometer signals was 0.005 (s) typically and 0.01 (s) at most (one sampling time).

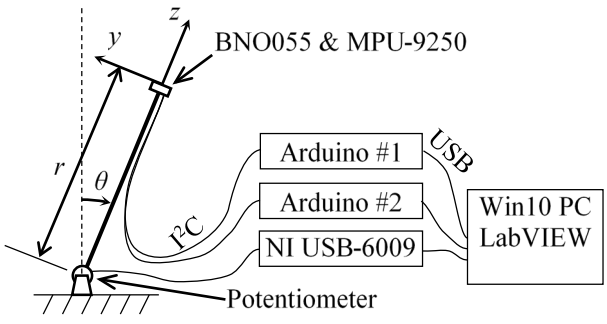

Figure 1. Inverted pendulum and Data Acquisition Setup. We mounted BNO055 and MPU-9250 on an inverted pendulum. Two Arduino Uno clones served as I^2C-to-USB/Serial translators. Data acquisition was triggered by data-ready signal of MPU-9250's DMP running at 100 Hz. The distance from the pivot point of the inverted pendulum to the IMUs was: r=0.5 [m]

2.2. Kalman Filter Implementation

We want to estimate the angular position of the inverted pendulum, θ, based on available raw IMU data: the angular velocity and linear accelerations. Ideally, time integration of the angular velocity should provide the

angular position, given an initial angle. In practice, however, such an approach would result in accumulation of errors, or drift. Taking advantage of known relationship between the available sensor signals, we implemented a Kalman Filter in order to estimate the angular position while reducing the drift.

The angular velocity measured by the IMU, $\hat{\omega}$, is modeled as the sum of the true angular velocity of the pendulum, $\dot{\theta}$, and some noise v_ω. The accelerations measured by the IMU, $\hat{a}_y$ and $\hat{a}_z$, include gravitational (g), transverse (y-direction), and radial (z-direction) components, as well as corresponding noises, v_y and v_z. We assume the noises are Gaussian White Noise. We know the constant distance from the pivot point of the inverted pendulum to the IMUs (r=0.5 [m] in our experimental setup). Considering simple kinematics of the pendulum, we then get the following equations.

$$\begin{bmatrix} \hat{\omega} \\ \hat{a}_y \\ \hat{a}_z \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ g\sin(\theta) - r\ddot{\theta} \\ g\cos(\theta) - r\dot{\theta}^2 \end{bmatrix} + \begin{bmatrix} v_\omega \\ v_y \\ v_z \end{bmatrix}. \quad (1)$$

Let us define the state vector as $\boldsymbol{x}_k \equiv [\theta \quad \dot{\theta} \quad \ddot{\theta}]^T$, and the system dynamics equations for discrete-time Kalman filter as follows:

$$\boldsymbol{x}_k = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \quad (2)$$

where the subscript k denotes the discrete time index, Δt is the sampling time, $\boldsymbol{G}$ is the input matrix, $\boldsymbol{u}$ is the input vector, and $\boldsymbol{w}$ represents the uncertainty of the system equations. Here we borrowed the symbolic notation from a tutorial by Rhudy et al. [9]. As we did not know or measured the $\boldsymbol{u}$ vector (torque acting on the inverted pendulum), we eliminated the term $\boldsymbol{G}\boldsymbol{u}_{k-1}$ from the equations. Defining system output (measurement) vector, $\boldsymbol{y} \equiv [\hat{\omega} \quad \hat{a}_y \quad \hat{a}_z]^T$, and measurement noise vector, $\boldsymbol{v} \equiv [v_\omega \quad v_y \quad v_z]^T$, we get

$$\boldsymbol{x}_k = \boldsymbol{F}_k(\Delta t)\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1} \quad (3)$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k \quad (4)$$

where $\boldsymbol{F}_k(\Delta t)$ is the system matrix as defined by Eq.(2), and $\boldsymbol{h}(\boldsymbol{x}_k)$ is the nonlinear observation function as defined by Eq.(1), which we linearized by calculating the Jacobian matrix, $\boldsymbol{H}$, at each time step:

$$\boldsymbol{H} \equiv \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} = \begin{bmatrix} 0 & 1 & 0 \\ g\cos(\theta) & 0 & -r \\ -g\sin(\theta) & -2r\dot{\theta} & 0 \end{bmatrix} \quad (5)$$

With this linearization, we have a form of Extended Kalman Filter, whose iterative algorithm is summarized as follows.

1. Define covariance matrices of the process and measurement noises, $\boldsymbol{Q}$ and $\boldsymbol{R}$ respectively
2. Initialize state vector, $\boldsymbol{x}_0$, and covariance estimate $\boldsymbol{P}_0$
3. Predict state vector: $\boldsymbol{x}_{k|k-1} = \boldsymbol{F}_k(\Delta t)\boldsymbol{x}_{k-1}$
4. Predict state error covariance: $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k\boldsymbol{P}_{k-1}\boldsymbol{F}_k^T + \boldsymbol{Q}$
5. Compute Jacobian by Eq.(5): $\boldsymbol{H}_k = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_{k|k-1}}$
6. Kalman gain: $\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T\left(\boldsymbol{H}_k\,\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}\right)^{-1}$
7. Update state estimate (using nonlinear observation function): $\boldsymbol{x}_k = \boldsymbol{x}_{k|k-1} + \boldsymbol{K}_k\left[\boldsymbol{y}_k - \boldsymbol{h}(\boldsymbol{x}_{k|k-1})\right]$
8. Update covariance: $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}$
9. Return to step 3 and continue.

In our experiment, we used Δt=0.01 [s], $\boldsymbol{Q}$=diag(0,10^{-6},10^{-5}), $\boldsymbol{R}$=diag(2×10^{-6},10^{-3},10^{-3}), $\boldsymbol{x}_0$=$[0,0,0]^T$, and $\boldsymbol{P}_0$=diag(1,1,1). Note that diag(a_1,a_2,a_3) denotes a diagonal matrix whose diagonal entries are a_1, a_2, and a_3.

2.3. Experimental Protocol

Using BNO055's "NDOF Fusion" mode, which calculates absolute orientation at 100 Hz from the accelerometer, gyroscope, and magnetometer data, we followed the instructions in the user's manual so that the device's auto-calibration status indicated "3" (i.e. "fully calibrated"; 0 indicates "not calibrated") in all four categories: system, gyroscope, accelerometer, and magnetometer. This calibration was done before every bout of experiment, when we temporarily detached the breadboard (with the IMUs on it) from the inverted pendulum and placed the breadboard on a flat desk surface. The calibration procedure for the accelerometer was to put the device in 6 different stable orientations, including the orientations in which gravity matched the device's x, y and z axis, for a period of a few seconds with slow transition between orientations. The calibration for gyroscope requires that the IMU be placed "in a single stable position for a period of few seconds", which was usually satisfied as a byproduct of calibrating the accelerometer. The calibration of magnetometer calls for "some random movements (for example: writing the number '8' on air)", which we did and made sure that the status indicated "fully calibrated". This typically took only a few seconds, drawing figure eight in the air just a few times. By far, the calibration of accelerometer was the most sensitive and time-consuming procedure. The status would jump around between 0 to 3 intermittently, taking minutes sometimes to get a stable "fully calibrated" status.

Once the BNO055 was calibrated, we reattached the breadboard on the pendulum. While the pendulum was upright (within ±0.2° per potentiometer reading) and stationary, the MPU-9250's accelerometer was zeroed at 0, 0, and 1g in x-, y-, and z-directions respectively. We enabled MPU-9250 gyro's optional auto-calibration feature, which triggered zeroing whenever several seconds of no motion was detected. We made sure that this happened at the beginning of every bout of experiment. We used the MPU-9250's "6-axis low power quartenions" mode, which integrated accelerometer and gyroscope signal by a proprietary Fusion algorithm built into the DMP software.

After calibration and zeroing of both IMUs, the inverted pendulum was manually moved back and forth in a swaying motion at various frequencies, while the LabVIEW program collected data. The data was exported to Excel spreadsheet, and then opened in MATLAB

(MathWorks, Inc., Natick, Massachusetts, USA) for analysis. We implemented the Extended Kalman Filter off-line in MATLAB. We also implemented the same algorithm in LabVIEW in real-time, which produced practically identical result only with small numerical errors. In the following sections, we present the result of MATLAB analysis.

3. Results and Discussion

3.1. Sample Data: Overall Observation

Figure 2 is an overview of sample data, where the manually imposed test pattern included about 20 seconds of no motion, followed by sweeping motions of about ±15-20° (t=20-60 sec), sinusoidal-like movements at 2-4 Hz (60-90 sec), and step-like motion (90-120 sec). Please note the Fusion and Kalman filtering had been running for a while and reached a steady state before we started data collection.

The black solid line denotes the potentiometer data as reference, i.e. a proxy of the actual angle of the pendulum. The blue color indicates data from BNO055, and the magenta color MPU-9250. The dashed lines indicate Fusion data, and the dash-dot lines the result of the Extended Kalman Filter. Additionally, the dotted lines indicate cumulative trapezoidal integration of the raw gyroscope data, which resulted in significant drift particularly with the BNO055. We also observed a large (~2.5°) offset in the BNO055 Fusion data. This offset value would vary from 0.5° to 3° typically at resetting and recalibration, was rather unpredictable, and persisted even though the calibration status indicated "fully calibrated". By the end of this test pattern, BNO055's calibration status of the magnetometer dropped from 3 to 2, but otherwise stayed at 3 (i.e. "fully calibrated").

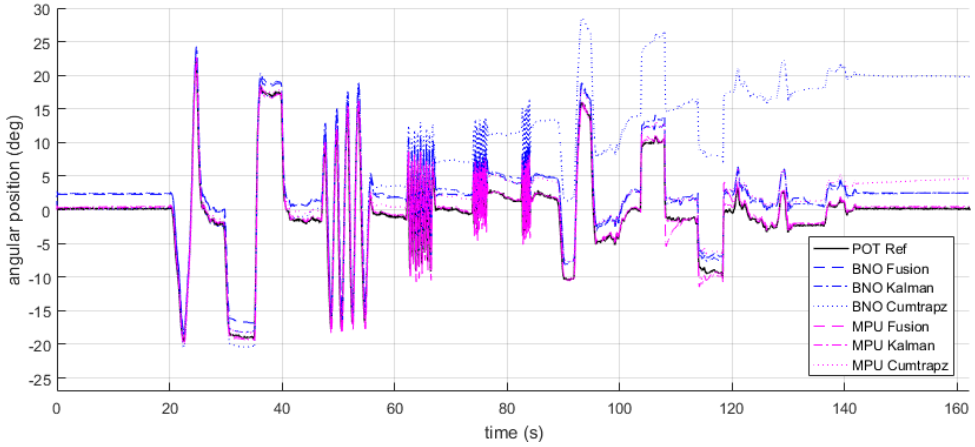

Figure 2. Sample Data Overview: Manually imposed motion included slow sweeping of about ±20°, sinusoidal-like swaying motion at 2–4 Hz, and step-like movements. Note significant drift in the straight time integration (Cumtrapz) of the measured angular velocities

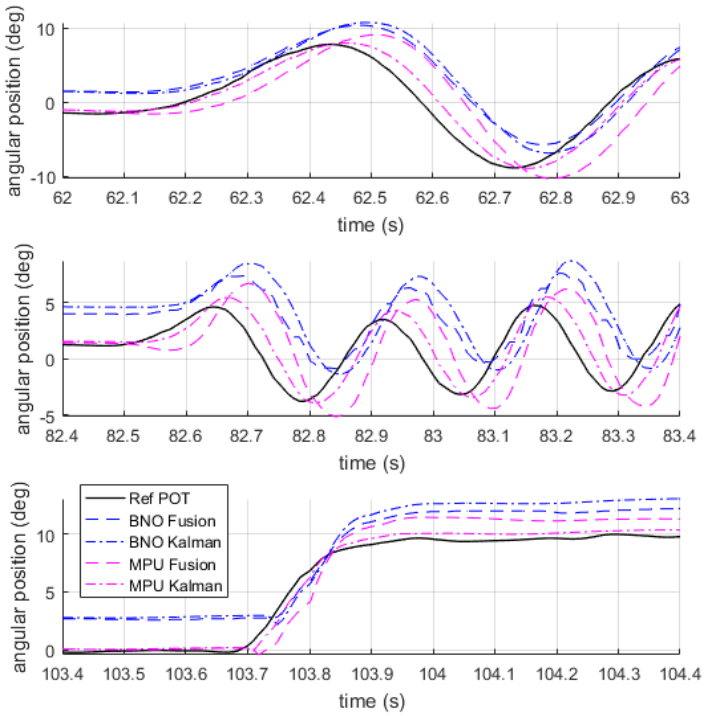

Figure 3. Sample Data Close-up View: Our implementation of Kalman Filter with the MPU-9250's raw data produced the best result overall. The BNO055 exhibited noticeably larger lag and offset. The BNO055 Fusion exhibited noticeable non-smooth fluctuations, most notably in the middle row (roughly 4-Hz sinusoidal motion)

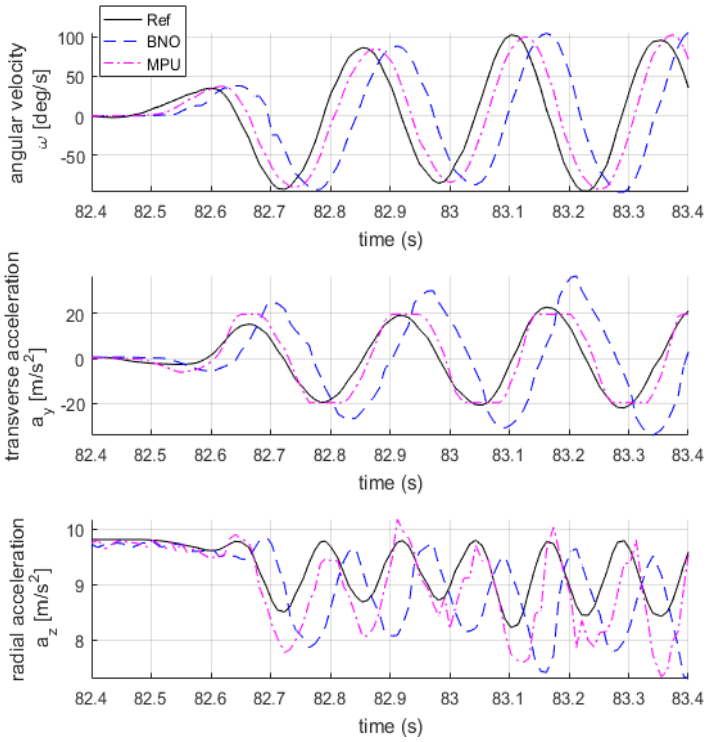

Figure 4. Raw sensor data corresponding to the onset of roughly 4-Hz sinusoidal-like motion (middle row of Figure 3, t=82.4–83.4). The BNO055 exhibited a larger time lag than the MPU-9250 did. Note MPU-9250's acceleration signal saturated at ±2g, which was the default full scale

3.2. Sample Data: Close-up View

Figure 3 shows a close-up view of three select sections of the same sample data, corresponding to Top) the onset of roughly 2-Hz swaying, Middle) the onset of roughly 4-Hz swaying, and Bottom) a sudden step-like motion.

The BNO055 Fusion produced relatively large lag and offset, 0.05-0.1 (s) and 2-3° respectively. It also exhibited curious fluctuations (most notable in Middle row, blue dashed-line). The Kalman Filtering on the BNO055 raw data resulted in similarly large lag and offset, without the fluctuations. The MPU-9250 Fusion performed better than BNO055 Fusion and Kalman Filtering of BNO055's raw signal. Our implementation of Kalman Filter with MPU-9250's raw signal resulted in the least amount of time lag and offset, roughly 0.02 (s) and 0.5° respectively, compared to the reference potentiometer data.

3.3. Sample of Raw Sensor Data

Figure 4 depicts the raw sensor data corresponding to the middle row of Figure 3 (t=82.4–83.4), which depicts the onset of roughly 4-Hz sinusoidal-like motion. The reference angular velocity was computed by taking finite difference (central difference method) of the angular position measured by the potentiometer. The reference transverse and radial accelerations were inferred by Eq.(1), while the angular acceleration $\ddot{\theta}$ was approximated by twice taking finite difference and g=9.816 [m/s^2].

The MPU-9250 showed better performance, with roughly 0.02 (s) of lag, than BNO055, which showed about 0.05 (s) of lag. This signifiant lag in BNO055's raw data explains the lag in its Fusion result.

We also noted peculiar flat spots and non-smooth behavior in BNO055's raw data: top row, t=82.7 [s], for example. We suspect this was caused by the proprietary algorithm related to the auto-calibration feature. It may have propagated through the proprietary Fusion algorithm, to produce the fluctuations observed in Figure 3.

4. Conclusion

Our initial observation of significant time lag in BNO055's Fusion output was quantified (0.05-0.1 sec). The time lag in BNO055's Fusion appeared to stem mainly from the lag in raw sensor signal. The lagging raw sensor signal of BNO055, when fed to our implementation of Kalman Filter, lead to a similar lag in the resulting angular estimation.

We noted unsmooth fluctuations in the BNO055's raw sensor signals and Fusion output, possibly due to the auto-calibration feature, which could not be disabled. The static offset in BNO055's Fusion output also varied, typically at resetting and recalibration, by an unpredictable amount (up to 2-3°).

In comparison, we observed better time response in MPU-9250's DMP Fusion output, likely due to more responsive raw sensor signals. With the responsive raw sensor signal of MPU-9250, our implementation of Kalman Filter produced the best performance overall, which supports our hypothesis. It is a promising option for our self-balancing bicycle project. As a next step, we plan to test these IMUs on an actual bicycle in motion.

Acknowledgements

The authors would like to thank Amritpal Sidhu and Russel (Gavin) Fielder, an Electrical Engineering major and an Applied Math major respectively at California State University, Chico, for their assistance in the project.

Statement of Competing Interests

The authors have no competing interests.

List of Abbreviations

IMU	Inertial Measurement Unit
MEMS	Micro-Electro-Mechanical Systems
DMP	Digital Motion Processor™ (by TDK Inven Sense)
NI	National Instruments.

References

[1] Cerone, V., Andreo, D., Larsson, M. and Regruto, D. "Stabilization of a Riderless Bicycle: A Linear-Parameter-Varying Approach", [Applications of Control] IEEE Control Systems, 30(5). 23-32. 2010.

[2] Finkbeiner, S., "MEMS for automotive and consumer electronics", Proceedings of the European Solid-State Device Research Conference, 9-14, Sept 2013.

[3] TDK InvenSense, "Embedded MotionDriver 6.12 - InvenSense Developers Corner", (log-in required) [Online]. Available: https://www.invensense.com/developers/software-downloads/. [Accessed May 2018].

[4] Kalman, R.E., and Bucy, R.S., "New Results in Linear Filtering and Prediction Theory," Journal of basic engineering, 83(1). 95-108. 1961.

[5] Leavitt, J., Sideris, A., and Bobrow, J.E., "High Bandwidth Tilt Measurement Using Low-Cost Sensors", IEEE/ASME Transactions On Mechatronics, 11(3). 320-327. 2006.

[6] "SparkFun Electronics," [Online]. Available: https://www.sparkfun.com/. [Accessed July 2018].

[7] "adafruit," [Online]. Available: https://www.adafruit.com/. [Accessed July 2018].

[8] Townsend, K, "Adafruit Unified BNO055 Driver," Adafruit Industries, [Online]. Available: https://github.com/adafruit/Adafruit_BNO055. [Accessed Feb 2018].

[9] Rhudy, M.B., Salguero, R. A., and Holappa, K., "A Kalman Filtering Tutorial for Undergraduate Students," International Journal of Computer Science & Engineering Survey, 8(1), Feb 2017.