Mobile Devices' Low Charge, High CPU Load, Number of Sensors, and OS Features Delay Sensor Data Readings

Boris Gafurov
Thinksense Inc.
North Glebe Rd, Suite 720
Arlington, VA 22203, USA
bgafurov @thinksense. com

Angelos Stravou
Thinksense Inc
North Glebe Rd, Suite 720
Arlington, VA 22203, USA
angelos @thinksense. com

***Abstract*—Sensor data reliability plays a critical role in applications that collect and use this data. The types of effects that can invalidate the data could be physical data drift, bias, missing data or outliers, to name a few. These can arise from several factors, where power management and operating systems, that are built to conserve energy and optimize application usage, are of the most concern. Thus, this study assessed an impact of the battery charge, functional state of the device, and the number of sensors employed. We used variability in the timestamps of the sensor data readings to measure the quality of the data acquired. Our experiments revealed that lower battery charge and high CPU loads have significantly impacted the mean interval between the sensor data readings. However while this happened at a very small scale (1-2ms) it had different impact for different devices. We also noticed that the number of sensors and device movement significantly changed the variability but not consistently across devices. Our most impactful finding is that sensor measurements can be significantly delayed, for up to 60ms. These large gaps in sensor readings can cause aliasing distortions affecting mobile apps. Moreover, our experiments show that the timing of sensor data can be inconsistent and highly dependent on the device's conditions. Detecting and compensating for sensor data loss and resampling the data, though at the cost of accuracy, may be a potential solution.**

Keywords— mobile devices, sensors, power management, data reliability.

I. Introduction

Sensors deployed in mobile devices collect environmental data at a certain sampling frequency in real time placed in a memory buffer for reading [1]. The most reliable way of obtaining continuous data is to read sensors with an app or a tool running in foreground. Unfortunately, having a foreground application or sensor collection is rarely an option as it interferes with the user experience. However, reading these data while in background state also presents its own challenges. For instance, when the sensors are managed by the Android OS, that is purposefully designed to conserve resources, the sensor raw data acquisition can be delayed. This can cause forward shifts in the data timestamps creating data causality inconsistencies. While that delay can appear random and variable, it can depend on many parameters [2]. In this study, we are focusing on how device's power management affects sensor data quality when running in background mode. While we can control the sensor data reading frequency, we do not have control over the wait queue scheduler for the buffer reading. Therefore, we do not have access to the time when sensor updates the buffer. However, in general, sensing is much faster (>100Hz) [3] than we will read the sensor data (5Hz). Hence, sensing frequency should not introduce significant differences in the timestamps by itself. In addition, we do have access to the lower chip-level sensing information through hardware API. This is a reasonable assumption for applications distributed through application markets [3,4].

Furthermore, we leverage the timestamp variability to estimate the reliability of the sensors data timestamps under different conditions and to understand better the sensing data quality. We assess data quality by evaluating the variance and extreme time intervals between captured sensor data. To that end, we use in-house designed mobile app to collect data from specific sensor(s) at a given polling rate. The app runs in the background during data collection while we are able to use the device. We show that there is a time variability between the sensor buffer readings and their delivery to the application caused by the operating system state and load. Thus, we quantify data timeliness quality for single and multiple sensor readings under normal conditions. We compare these measurements with the ones we collect when the device exhibits low power and high CPU utilization or heavy user's activity. As exemplary sensors we selected the accelerometer and gyroscope sensors, which are most commonly used for health and other research [4-7]

II. Methodology

In our experiments, we chose Google Pixel 6a and Samsung S22 smart phones equipped with Android 11 operating system. These are modern and powerful devices with a variety of sensors from which we can capture data. They also have an advanced power management system and should be ideal for the experiments we are planning to conduct. Data from sensors will be acquired in two modes, when device is stationary or moving, with the latter meant to provide more realistic data sets. Device's displacement will be accomplished using Elephant robotic arm myCobot 280 using the same trajectory stored in the arm's memory

A. Single-sensor study

The *first operating condition* is 100% charged but not plugged-in device. This condition provides "pristine" conditions and establish a relatively consistent data collection as a baseline. We expect that "pristine operations" should reveal timestamp intervals variability that one should expect during a daily use of their devices. In real-world 100% charged battery is quite rare and it does not last for very long, However, in our experiments, we use this as a baseline to weed-out normal data sensing collection variability when compared to low charge devices.

The *second operating condition* is 10% battery charge remaining and the device is not plugged-in. This imitates data

979-8-3503-5909-1/24/$31.00 ©2024 IEEE

recordings under the various power conserving mechanisms utilized by the device's hardware and OS. As expected, the second condition produces the greatest variability in the readings that will provide insight into power management control of and access to the sensor's data.

The *third operating condition,* 100% charged device under a heavy, nearly 100%, computational load. This running environment exposes how the device's load management systems enforce scheduling and pre-emption of task threads depending on their power and CPU consumption. In this experiment, our app CPU tasks and power consumption are negligible compared to the load we induce. Our results reveal how the OS consumption profiles implemented and the users' activity impact the quality of the sensor data.

B. *Multi-sensor study*

In the next set of experiments, we simultaneously record data from two sensors using the aforementioned recording conditions. The variability of timestamps in accelerometer and gyroscope sensors will be assessed. This data will elucidate the impact on reliability of reading from multiple sensors in various conditions. In addition to the sensor readings variability we can assess their cross-correlation (lag). If in fact it is pronounced and increased or decreased that would indicate that there is a relationship that a hardware and/or power management introduces to multi-sensor data readings.

C. *Data and Statistical analyses*

Data on variability expressed as means and standard deviations can be compiled from our experiments and statistically assessed for significant differences. The variability of the mean interval between sensor readings across different acquisition conditions will be analyzed for the significant changes using 2-tailed t-test function. To show significant changes in the distributions of the intervals between sensor readings, data will be analyzed using two sample Z-score test formula written in Excel. 3D sensor data converted into a simple geometrical distance or rather acceleration or rotational speed for gyroscope absolute value would be routinely tested against the intervals between sensor readings for cross correlation to ensure that there are no underlying relationships between them. Cross correlation function (CCF) vs. lag relationships will be calculated with no data wrapping and plotted out in Excel using custom written macros. Throughout the text of this manuscript averages will be expressed as a mean ± standard deviation (SD) followed by number of measurements in parentheses.

D. *Application design*

Our app to collect sensor data was developed in Android Studio (v.2022.1.1). Briefly, app is designed to listen to single or multiple sensors without buffering every 200ms or 5Hz polling rate. When sensor OnSensorChanged listener receives the input from the sensor, app checks if the time, since the last data was recorded for this sensor, is greater than our default polling of 200ms, then x, y, z metrics of the sensor data are stored in local database. To make OS not to suspend our app, when it goes into background, or interfere with the app functionality depending on the battery status, the Wake Lock mechanism was implemented and battery optimization was disabled in the app. Experiments under each condition were run for 5min, that was determined to be the maximum time during which battery percentage did not change under the high CPU load. Middle 3min portion of that was used for further off-line analyses.

III. RESULTS

In general, intervals between the reading were very close to the polling rate of 200ms. Despite that, variability of the intervals between sensor data readings showed significant differences in means across all the conditions, albeit a small one, on a scale of 0.1-0.2ms, perhaps due to the great number of data points. Although, most importantly, under certain conditions, intervals showed whopping, however intermittent, up to 60ms increase. So, despite the Wake Lock being engaged there are still times where it fails or pauses continuous app operations. Moreover, occurrence of long intervals showed opposing tendencies between the sensors tested. These great outliers, ultimately, changed statistics between the interval distributions. Their nature may require further study. CCF analyses between the intervals and the values from sensor readings did not show correlation, normally R was less than 0.1, meaning that sensor activity itself did not, as expected, impose significant stress on sensor functionality [1].

A. *Google 6a phone*

Accelerometer study has revealed that when Google Pixel 6a device was 100% charged and stationary intervals between the sensor data reading were significantly different from 10% and 100% charged with CPU working at a high load (Fig.1 A vs C and E). These results were based on the comparisons of both interval means with t-test and interval distributions with Z-test. Moving device showed significant difference only between 100% charged and high CPU load cases, again in both interval means and distributions (Fig.1 B vs F). Moving vs. stationary comparisons were significant only in 10% charged device and only in interval distributions (Fig.1 C vs D).

Again, we acknowledge that the absolute differences between the means were small on a scale of the 0.1-0.2ms (e.g., Fig.1 A, C, and E showed 200.35±0.58ms, 200.46±0.6ms, and 200.43±0.52ms, respectively) but significant, as previously mentioned. Nevertheless, we are documenting all of them here and in the sections below, so all the data analyses are available to the readers.

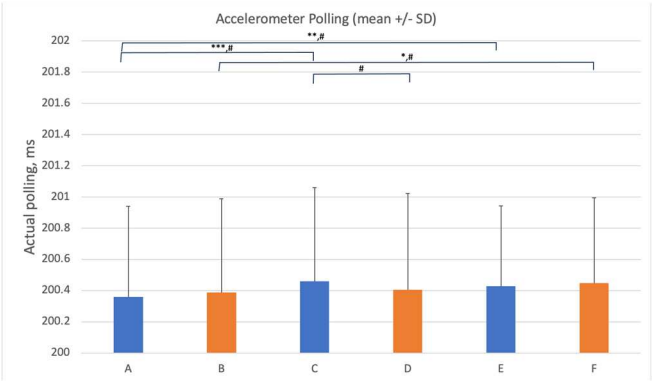

Fig. 1. 100% charged Pixel6a shows slightly lower intervals between the sensor readings. * - significant mean difference obtained by t-test (* - p<0.05, ** - p<0.01, *** - p<0.001, n=898). # - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05). Blue A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. Orange B, D, F – same conditions but when device was moving.

Closer inspection of the sample interval distributions (Fig.2) showed occasional longer intervals up to 9ms in duration (10, in total, points outside of upper extreme of the

whisker box plot). In general Google 6a accelerometer data showed longer intervals among all but one condition.

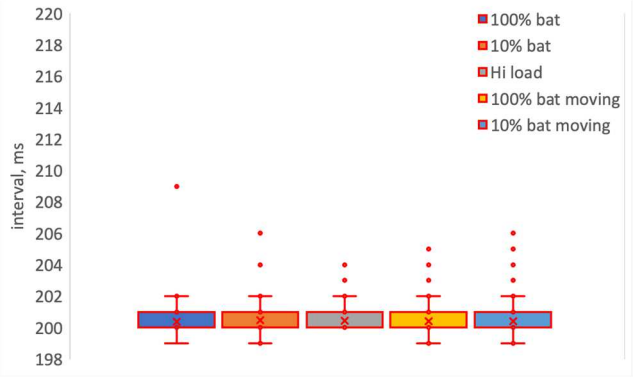

Fig. 2. Interval distributions of accelerometer sensor data access in stationary and moving Pixel6a. Hi load CPU moving device did not show long intervals (not show).

Gyroscope study, unlike accelerometer one, mostly showed significant differences in moving devices and not the stationary ones (Fig.3). Stationary devices showed no significant differences in means between different conditions. The only statistical difference was detected between interval distributions in 100% charged vs. high CPU load comparison (Fig.3 A vs. E). However, there were differences between stationary and moving devices in 100% charged and high CPU load cases both in means and interval distributions (Fig.3 A vs. B, E vs F). On the other hand, moving devices revealed that intervals between the sensor readings in 10% charged device were significantly different from 100% and high CPU load cases (Fig.3 D vs B, D vs F). That tendency was just the opposite to the Accelerometer study.

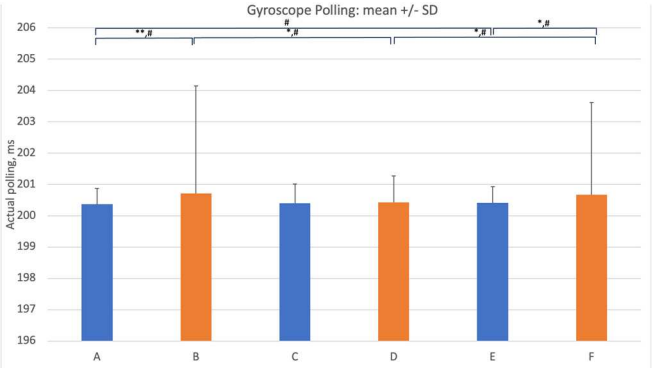

Fig. 3. Gyroscope data reading intervals were slightly higher when Pixel6a was moving. * - significant mean difference obtained by t-test (* - p<0.05, ** - p<0.01, *** - p<0.001, n=898). # - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05). Blue A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. Orange B, D, F – same conditions but when device was moving.

To get an insight into low battery charge impact on the longer intervals we plotted out the distributions for all experiments where they were detected (Fig.4). It is clearly can be seen that high SDs in 100% charged and high CPU load were caused by occasional long up to 58ms intervals, i.e., Figure 3B, D, and F showed 200.7±3.4ms, 200.4±0.9ms, and 200.7±3.0ms, respectively. Long intervals appeared in gyroscope sensor data much more often 23 vs 10 in accelerometer and not as uniformly. Also, two stationary conditions (100% charged and Hi load) did not show longer (than upper extreme) intervals.

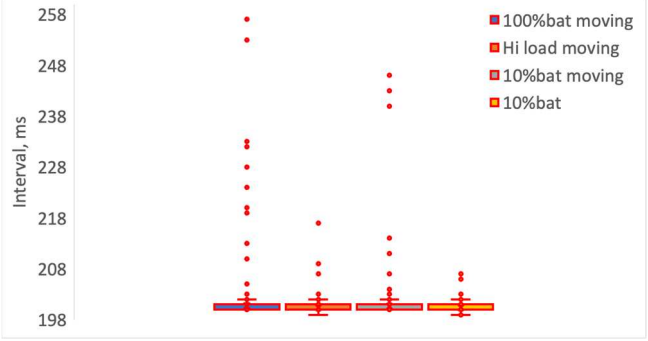

Fig. 4. Interval distributions of gyroscope sensor data access in stationary and moving Pixel6a. 100% charged and Hi load CPU stationary devices did not show long intervals (not show).

Accelerometer and Gyroscope study, when two sensors were simultaneously recorded from, revealed no significant differences in means in the contrast to the single sensor recordings (Fig.5). However, there were significant differences between the distributions of the intervals. Firstly, there was significant difference in interval distributions between stationary and moving devices when they were 100% charged for both accelerometer and gyroscope, and at high CPU load but only for accelerometer. There was no difference between the cases when device was stationary. However, there were differences when device was moving but only between 10% charged device and both 100% charged and high CPU load for the accelerometer reading intervals. For gyroscope, the difference was detected only when compared with high CPU load. The later observation was reminiscent of the one that we saw in gyroscope alone study, although not for the accelerometer. Somehow, accelerometer data acquisition began acting similar to gyroscope, or perhaps it was influenced by it. But this observation needs further investigation.

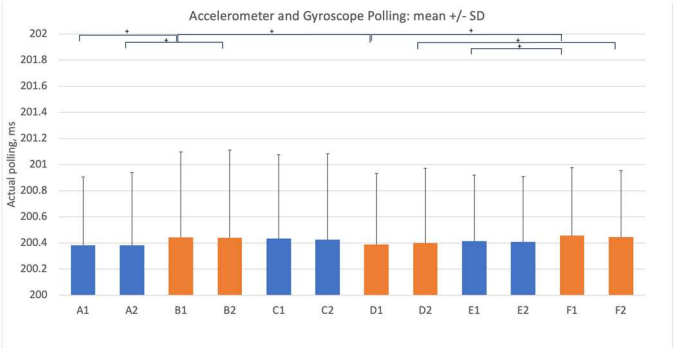

Fig. 5. Mean intervals between data reading recorded from Pixel6a accelerometer and gyroscope simultaneously. ± - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05). Blue A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. Orange B, D, F – same conditions but when device was moving. Indexes 1 or 2 denote Accelerometer or Gyroscope sensor readings, respectively.

The whisker plot of distributions revealed that longer up to 10ms intervals were again present but to the moderate extent of 15 in total, with 7 for accelerometer and 8 for gyroscope. It seems that these long intervals always appear, however their number depends on sensor type. Moreover, it

depends on the number of sensors read, because they appeared more uniformly under the current conditions.

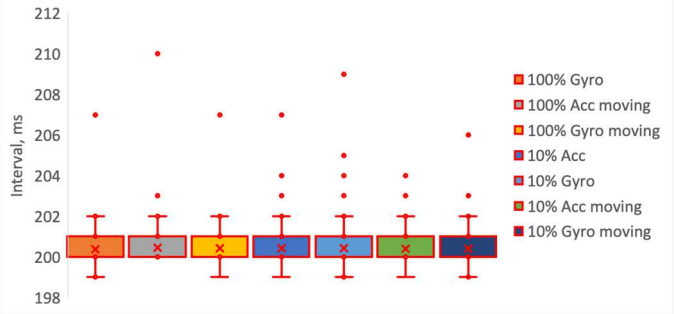

Fig. 6. Interval distributions of accelerometer and gyroscope sensor data accessed simultaneously in stationary and moving Pixel6a. 100% charged Accelerometer in stationary device and either of Hi Loads did not show long intervals (not show).

Since in this study we collected pairs of data sets, i.e. intervals between sensor data access for both gyroscope and accelerometer, in the course of one experiment, we can investigate if there is any degree of relationship between them. Fig.7 shows cross-correlation vs lag plot for each condition for moving or stationary device. The highest and very distinctive, though low to be considered as significant (r=0.2), correlation peak appeared for stationary device when battery was at 10%. Thus, there seems to be a distinct interplay between the sensors, however statistically insignificant.

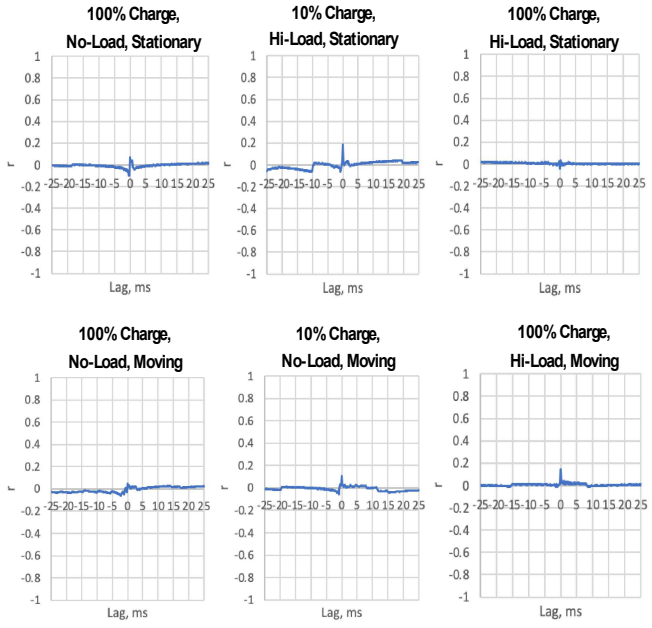

Fig. 7. Cross-correlation analyses of accelerometer vs. gyroscope intervals in stationary and moving Pixel6a. Cross-correlation between gyroscope and accelerometer reading intervals showed weak interplay when device was 10% charged and stationary.

So, it clearly can be seen that multi-sensor reading introduced changes in the data collected on Google Pixel 6a device. Whether it is due to the application design or hardware implementation will require further investigation.

B. *Samsung S22 phone*

To investigate differences between manufactures, we have collected the same as above data sets on a different, but comparable to Google Pixel 6a by its specs, Samsung S22 device. Results as described below appeared to be quite different between the different manufactures despite that we made all the efforts to keep environment as similar as possible, i.e. same OS version, same set of default applications, etc. The following figure presents the data for Samsung S22 in the same fashion as for Google Pixel 6a.

Accelerometer study revealed that sensor data collected when phone was 100% charged and operating at high CPU load were highly and significantly different from both 100% and 10% charged phones judging by the means and interval distributions (Fig.8). Mean values were different by up to 2ms, as seen on a Fig.8 (e.g., Fig. 8 A, C, and E showed 201.9±1.8ms, 202.0±2.2ms, and 200.9±0.9ms, respectively).

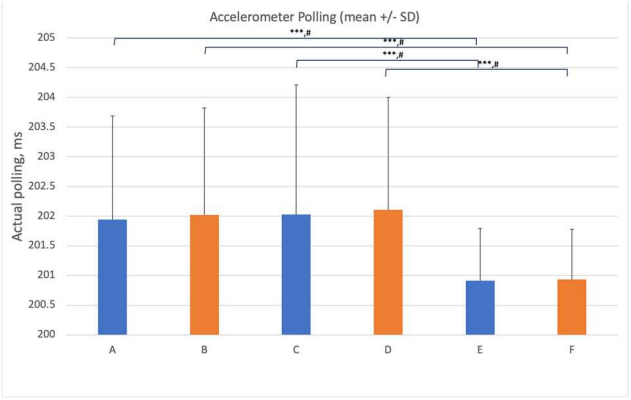

Fig. 8. SamsungS22 accelerometer data reading intervals were the lowest at high CPU load. * - significant mean difference obtained by t-test (* - p<0.05, ** - p<0.01, *** - p<0.001, n=898). # - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05). Blue A - 100\% charged battery, C – 10\% charged battery, E – 100\% charged battery with CPU at high load when device was stationary. Orange B, D, F – same conditions but when device was moving.

The whisker plots (see Fig.9) showed existence of the longer intervals with 11 in total and with up to 36ms in duration. This was comparable with Pixel 6a device in numbers, durations, and uniformity, except for the 10% charged condition, where very long (36ms) interval was documented.

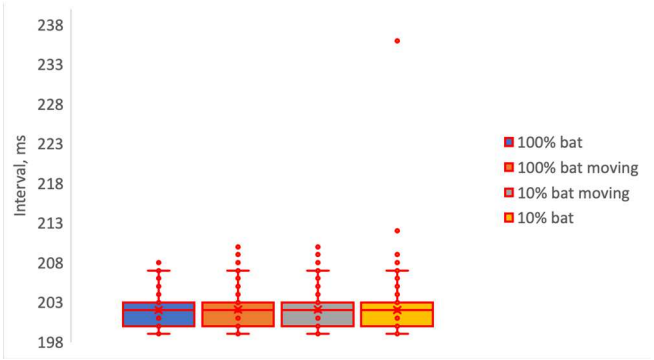

Fig. 9. Interval distributions of accelerometer sensor data access in stationary and moving SamsungS22. Hi load CPU both stationary and moving devices did not show long intervals (not show).

Gyroscope study on Samsung S22 phone revealed similar tendencies, i.e. the high CPU load showed the lowest and

highly significant values by means (around 1ms, i.e., Fig.10 A, C, and E showed 202.0±1.8ms, 202.0±1.9ms, and 201.0±2.0ms, respectively) and interval distributions (Fig.10). Moreover, there was a significant difference between stationary and moving devices in high CPU load case (Fig.10 E vs F).

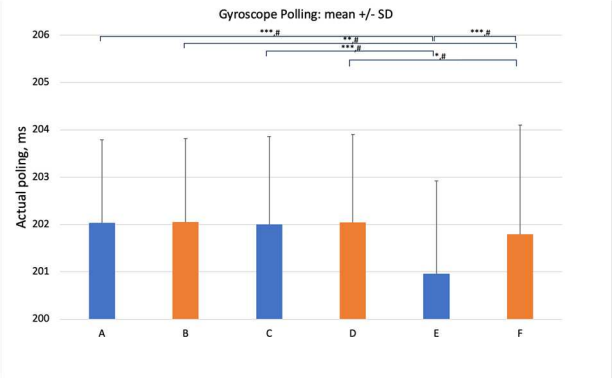

Fig. 10. SamsungS22 gyroscope data reading intervals were the lowest at high CPU load. * - significant mean difference obtained by t-test (* - p<0.05, ** - p<0.01, *** - p<0.001, n=898). # - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05). Blue A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. Orange B, D, F – same conditions but when device was moving.

Whisker plots (see Fig.11) showed existence of longer intervals, 16 in total, with up to 52ms in duration. This was not the same in numbers and durations when compared with Pixel 6a device that showed substantially greater number of long intervals and in different conditions.

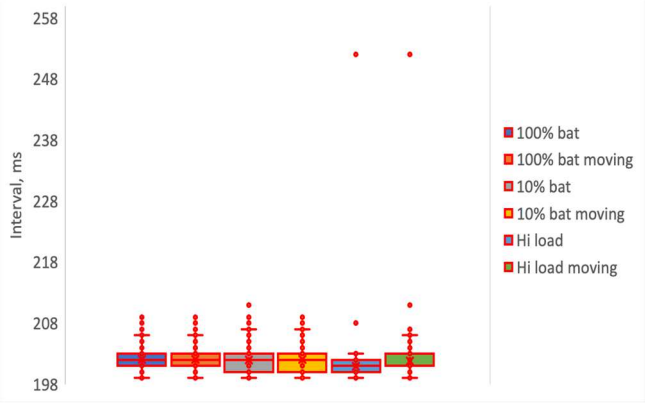

Fig. 11. Interval distributions of gyroscope sensor data access in stationary and moving SamsungS22.

Accelerometer and Gyroscope study in general produced very analogous results as single sensors studies (Fig.12). Namely, data obtained for the high CPU load, however only for stationary devices, were highly significant for 100% and 10% charged devices. Moreover, data from 100% charged moving device showed significant change in the distribution when compared with high CPU load (Fig. 13).

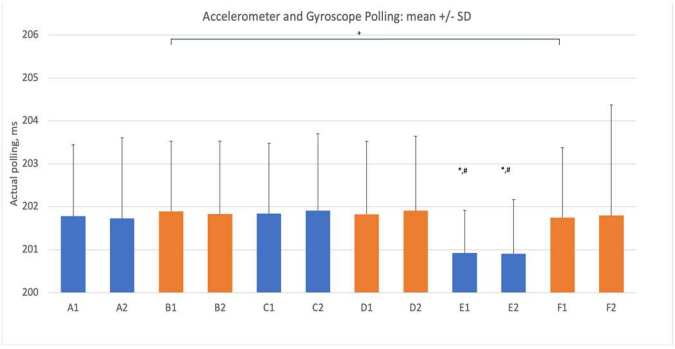

Fig. 12. Accelerometer and gyroscope data reading intervals were the lowest at high CPU load in stationary SamsungS22. ± - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05). * - significant mean difference obtained by t-test (p<0.001, n=898) between E1,2 and A1,2 or C1,2. # - significant difference in interval distributions obtained by Z-test (score > 1.645, alpha < 0.05) between E1,2 and A1,2 or C1,2. Blue A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. Orange B, D, F – same conditions but when device was moving. Indexes 1 or 2 denote Accelerometer or Gyroscope sensor readings, respectively.

The whisker plot of distributions revealed that longer up to 60ms intervals were again present but to the moderate extent of 17 in total, with 6 for accelerometer and 11 for gyroscope. Main difference from Pixel 6a device was the relatively higher duration of these long intervals in Samsung S22 and the presence of much longer, up to 60ms interval in Hi load CPU conditions in gyroscope sensor data.

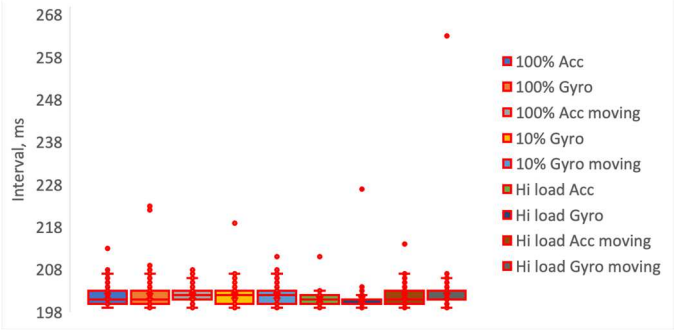

Fig. 13. Interval distributions of accelerometer and gyroscope sensor data accessed simultaneously in stationary and moving SamsungS22. Gyroscope in 100% charged moving devices, and Accelerometer in 10% charged stationary and moving devices did not show long intervals (not show).

Clearly, Samsung S22 phone showed the same tendency in average polling rates in both multi and single sensor data acquisitions. Goggle Pixel 6a phone, on a contrary, did not show that. Moreover, occurrence of the longer intervals was found to be different in both devices. Perhaps that may have been caused by the differences in the hardware design of these devices.

To be consistent with Google Pixel 6a phone study we investigated if there had been any degree of relationship between gyroscope and accelerometer data. Figure 14 shows cross-correlation vs. lag plot for each condition for moving or stationary device. In contract to Google Pixel 6a that had highly distinctive peak with r=0.2 and various others around 0.1, Samsung S22 did not show the same or anything close to that. All cross-correlations vs. lag plot were rather unimpressive. That may account for the lack of any relationships between gyroscope and accelerometer (as it should be) in Samsung, however Google might have

displayed some background interplay between the sensors (but it should not have).

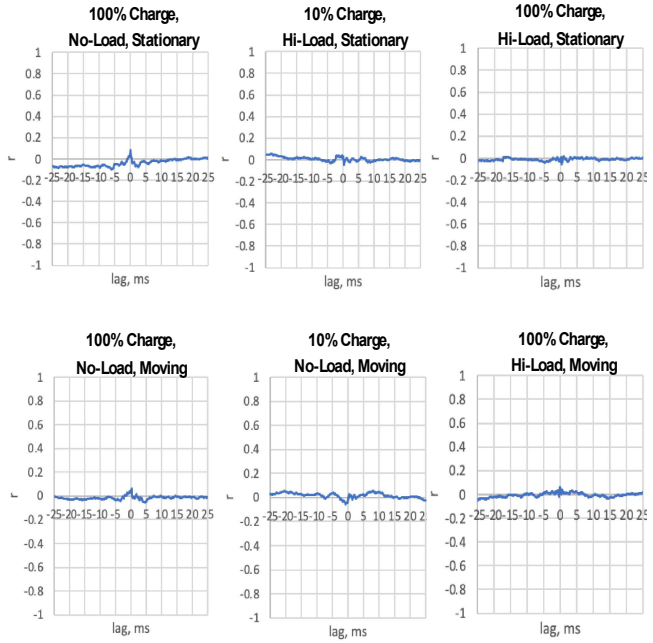

Fig. 14. Cross-correlation analyses of the accelerometer vs gyroscope intervals in stationary and moving SamsungS22. Cross-correlation between gyroscope and accelerometer reading intervals showed they were accessed mostly independently.

IV. Discussion

In our experiments, we collected sensor data under specific and controlled operating conditions. Our results showed that even under these controlled conditions, we were able to easily identify deviations for sensor readings that can potentially frustrate applications that rely on machine learning algorithms [8-10]. We can easily see how, for more complicated tasks, especially for medical research or day-to-day life activities, sensor's data reliability is becoming very important. Unfortunately, many factors can interfere with the data quality collected by mobile phone data sensors. More specifically, we documented two, generally expected challenges. One was the measurement variability due to compute and processing delays. Another, even more surprising but much more severe (up to 30% of the polling rate) delay due to power saving strategies imposed by the OS battery conserving policies.

A. Average polling rates

The Pixel 6a device showed on average about 0.5ms difference from our demanded 200ms polling period. The Standard Deviation (SD) ranged from 0.5-1.0ms. Surprisingly, the Samsung S22 measurements had even longer 1-2ms difference from that 200ms polling period with SD ranging from 1-2ms. These delays were determined by the processing power of the devices showing that Pixel 6a was 2 times faster than Samsung S22.

The absolute differences between the means caused by the different conditions were small on a scale of the 0.5 and 2ms, for Pixel 6a and Samsung S22, respectively, however significant, as it has been acknowledged above. At our 200ms polling period these were negligible, though at a faster data acquisition, to achieve greater accuracy about the device and/or owner activities [11], might play a crucial role.

On the other hand, these times are on the scale of jitter effect in smartphones that seem to have a significant impact on our data [13,14,17]. Perhaps, one of the explanations could be because sometimes they were accessed from the cache and not actually from the memory buffer that is managed by PMI and OS.

B. Long between reading intervals

Based on our extensive sensor data collection, we have demonstrated that there exist substantial measurement gaps between sensor readings of up to around 60ms. They did not seem to be related to the physical state of the device, but they were influenced by the number of sensors read, at least in Pixel 6a. We posit that this had to mostly with the Pixel 6a OS features. Whether the actual x, y, and z metrics of accelerometer and gyroscope were reflecting real device movement could not be answered by this study. However, that brought a question if, or to what extent this intermittent paucity can affect research that depends on data consistency, e.g., signaling analysis, continuous monitoring, etc. From the Nyquist sampling theorem to avoid aliasing one must ensure at least two fold acquisition frequency over the possible bandwidth of the signals. The observed 60ms gaps and the decrease in sampling frequency, however intermittent, can drastically reduce the effective sampling frequency to 8.3Hz. This sensor data acquisition limitation may impact high frequency data analyses where the bandwidth of the sampled signals may drop suddenly all the way down to about 8Hz. This can subsequently lead to a missing signals or activity, and essentially introduce erroneous lower frequency activity.

Furthermore, if the application is tracking the movements and visualizing them via a world coordinate system transformation for acceleration, trajectories will drift off course. This is due to longer accelerations at certain times. As reported by authors [12], acquisition for detecting subject's movement ranges from 20Hz-200Hz. With 60ms extra, it easily may, even for the lowest 20Hz frequency, double the time subject spent ac/de-celerating at selected instances that would quadruple the displacements. Authors in [12] actually reported over estimation limitations in their research that might as well have been caused by those extra-long intervals. High-pass digital filter was proposed as a remedy to attenuate lower part of the spectrum but it only would minimize it. So, technically, one needs to discard this point as a continuous and devise a reset point for trajectory integration routine or resample the data set.

When reviewing available online open-source sensor data sets (e.g., [15,16]) one can see that data were collected with the applications being in foreground on a smartphone, i.e., app was kept in background but to report a behavior it had to be brought up, and for a short period of time 3min or 20sec with relatively high acquisition frequencies of 40-50Hz. These data sets were very consistent and did not have gaps in data. We calculated the intervals between the data timestamps and were very surprised to see virtually constant interval representing acquisition frequency throughout whole 20s recording session [16]. This could not be replicated by our app, even if it ran in the foreground. Clearly, the app design is different and simply collects raw data at the fastest sensor refresh rate (50Hz), however, for a short period of time. Later, we assume, data

were resampled to the desired rate of 40Hz for the later analyses. Our app conserves activity using onSensorChange method with preset polling period limit that is more suitable for continuous applications. Ultimately, in our case, that introduced variability and long gaps between the sensor data readings. Another approach to this limitation would be switching to collecting raw timestamps from the sensor itself and not from the OS. This task is technically difficult because raw sensor timestamps are not uniform and counted from the specific time epoch, so it does not specifically reflect the time when the event happened. That feature is set by the manufacturer, so it too may vary from device to device.

V. Conclusion

The number and quality of mobile device sensors have increased over the last few years. In this study, we exposed some of the data quality issues observed when the device is under different operating conditions. Indeed, the battery charge, functional state of the device, and different device hardware can introduce unwanted data inconsistencies and jitters when capturing sensor information. Moreover, the Operating System (OS) can under certain conditions delay the sensor data acquisition. These delays can cause aliasing at higher than 8Hz frequency ranges and affect high-frequency sensing. In addition, we observed that mobile devices from different manufacturers and the number of sensors monitored may exacerbate time gaps in sequential sensor data. Finally, due to the data delays, any applications that require data quality guarantees will have to implement data validation and/or resampling procedures. This is essential before data can become usable for applications and research that require high sensing frequencies and data consistency.

References

[1] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, A survey of mobile phone sensing. IEEE Communications magazine, vol. 48, no. 9, pp. 140–150, 2010.

[2] Stisen A., Blunck H., Bhattacharya S., Prentow T.S., Kjærgaard M.B., Dey A., Jensen M.M. Smart devices are different: Assessing and mitigating gmobile sensing heterogeneities for activity recognition. Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems; Seoul, Republic of Korea. 23 November 2015; pp. 127–140.

[3] G. Grouios, E. Ziagkas, A. Loukovitis, K. Chatzinikolaou, E. Koidou. Accelerometers in Our Pocket: Does Smartphone Accelerometer Technology Provide Accurate Data? Sensors (Basel). 2023 Jan; 23(1): 192.

[4] J.R. Kwapisz, G.M. Weiss, and S.A. Moore. Activity recognition using cell phone accelerometers. SIGKDD Explorations Newsletter, 12(2):74–82, 2011.

[5] T. Brezmes, J.L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, pages 796–799, 2009.

[6] W. Wu, S. Dasgupta, E.E. Ramirez, C. Peterson, and G.J. Norman. Classification accuracies of physical activities using smartphone motion sensors. Journal of Medical Internet Research, 14(5), 2012.

[7] Douangphachanh V., Oneyama H. Exploring the use of smartphone accelerometer and gyroscope to study on the estimation of road surface roughness conditions. Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics; Vienna, Austria. 1–3 September 2014; pp. 783–787.

[8] Zhai Y., Nasseri N., Pöttgen J., Gezhelbash E., Heesen C., Stellmann J. Smartphone accelerometry: A smart and reliable measurement of real-life physical activity in multiple sclerosis and healthy individuals. Front. Neurol. 2020;11:688.

[9] Culhan K., O'Connor M., Lyons D., Lyons G. Accelerometers in rehabilitation medicine for older adults. Age Ageing. 2005;34:556–560.

[10] Chinrungrueng J., Sartsatit S., Intarapanich A. My act: An automatic detection of daily physical activity and calorie expenditure using smart phones. J. Assist. Rehabil. Ther. Technol. 2014;2:23187.

[11] Yurur, O., Labrador, M. & Moreno, W. Adaptive and energy efficient context representation framework in mobile sensing. IEEE Trans. Mob. Comput. 2014; 13, 1681–1693.

[12] Straczkiewicz, M., Huang, E.J. & Onnela, JP. A "one-size-fits-most" walking recognition method for smartphones, smartwatches, and wearable accelerometers. npj Digit. Med. 2023; 6, 29.

[13] E. Peguero, M. Labrador and B. Cook, Assessing Jitter in Sensor Time Series from Android Mobile Devices. 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, USA, 2016, pp. 1-8.

[14] W. Ramadan, E. Ozer. Modal analysis under jittering and kernel clock distribution: single-output identification. 2023. Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction.

[15] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, Fusion of smartphone motion sensors for physical activity recognition. Sensors, vol. 14, no. 6, pp.10146–76,2014.

[16] Vaizman, Y., Ellis, K., and Lanckriet, G. Recognizing Detailed Human Context In-the-Wild from Smartphones and Smartwatches. IEEE Pervasive Computing, vol. 16, no. 4, October-December 2017, pp. 62-74. doi:10.1109/MPRV.2017.3971131

[17] M. Benndorf, M. Kaden, F. Ringsleben, C. Roschke, R. Thomanek, M. Gaedke, T. Haenselmann. Investigating the Influence of CPU Load, Memory Usage and Environmental Conditions on the Jittering of Android Devices. ICNCC'18: Proceedings of the 2018 VII International Conference on Network, Communication and Computing December 2018, pp. 102–106 doi.org/10.1145/3301326.330