Sensor fusion on redundant angular rate sensors

Author: Marcos Díaz

Research period: 2021-2022
Publication: November 2022

Abstract

It Is possible to improve angular rate sensor performance by combining data from multiple redundant sensors. Noise can be reduced by sampling each sensor in parallel, zero-rate can be reduced by opposing bias cancelling each other, and range can be extended without tradeoffs by configuring each sensor to different scales.

Acronyms

ARS: Angular Rate Sensor.
DPS: Degrees Per Second.
DTG: Dynamically Tuned Gyroscope.
GPS: Global Positioning System.
HID: Human Interface Device.
IMU: Inertial Measurement Unit.
PDF: Probability Density Function.

1 Introduction

Redundant sensors (with equivalent characteristics) used in critical systems (aviation, medical) increase the reliability of the system in case of failure, but the individual characteristics of each sensor remain the same. [1]

In the other hand, is common to find multiple sensors of different nature used in consumer electronics, complementing each other to obtain better results (sensor fusion). An example of this is a smartphone combining **GPS** data and motion sensor data to reduce the uncertainty of location. [2]

We will evaluate the possibility of using sensor fusion with redundant angular rate sensors to improve sensor performance (noise ratio, zero-rate and scale) in the domain of computer input.

2 Problem

2.1 Limitations of Angular Rate Sensors

As for today, most angular rate sensors available belong into two groups:

Consumer electronics **IMU**s, sometimes with both accelerometer and **ARS** in a single chip, used in devices such as smartphones and videogame peripherals. These are based on the piezoelectric effect, are small, relatively inexpensive, and are energy efficient. Due to their small size, its performance characteristics (noise, zero-rate, scale) are limited, but enough for its intended uses such as smartphone tilt detection, step-counter, and mobile videogame input. [3]

The next tier of motion sensor devices are industrial-grade or aerospace-grade, such as ring laser gyroscopes (Sagnac effect) and **DTG** rotor gimbals [4]. These achieve very precise readings but are expensive and big, not suited for consumer electronics.

Smartphone-grade IMUs prioritize small footprint and low energy consumption in their design, while performance characteristics such as noise ratio and zero-rate (drift) are only secondary. The miniaturized parts featuring the piezoelectric effect are prone to have high relative noise (compared with other common input methods such as a mouse optoelectronic sensor); and small variations in temperature cause an offset in their readings within a noticeable range.

One of these novel application that require higher motion sensing performance than the provided by low-tier IMUs is Virtual Reality in the domain of videogames, in which aiming a (virtual) weapon is usually done by using both hands, each hand having an independent device featuring infrared spacial tracking, so the aim angle can be derived from the position of the hands in 3D space. On the contrary, aiming a pistol with a single hand must rely much more in single device angle detection, with much lower accuracy.

Videogame controllers (Sony PlayStation DualShock, Steam Controller, Nintendo Switch Joy-Cons, etc...) have also been featuring motion sensors, but they were never used as a primary method for aiming (or other application that require very precise angular movements detection) due to the limitations of these smartphone-grade IMUs; and instead only using angular rate data as secondary input method to complement thumbstick or trackpad primary input for aiming.

2.1.1 Noise

Noise is the unintended random disturbance on a signal. In commercially available ARSs noise is measured in $mdps/\sqrt{Hz}$ (milli-degree per second by square root of frequency).

Is not trivial to do a fair comparison with other common computer input methods such as mouse optoelectronic sensors, since these are a combination of camera and image processing hardware/software that effectively reduces noise to zero on the resulting output. [5]

In practical terms, when using a optoelectronic sensor mouse for controlling a computer mouse cursor, noise is a non-factor; but when using a commercially available ARSs (using a reasonable **DPS** scale and operative system sensitivity) the noise can cause the mouse cursor to jitter several pixels back and forth at high frequency; making it unfit for the purpose unless applying filtering techniques with significant tradeoffs in latency (smoothing) or linearity (acceleration).

Common solutions for reducing noise are:

- Frequency filtering: Either via hardware or software, to reduce the amplitude of undesired frequencies. Usually removing frequencies that are much higher or much lower than what the user is expected to input as real signal. By definition cannot remove noise in the frequency range of real input, or it would be filtering out such input.
- Multisampling: Since noise is a random value within a range, averaging the value of samples over time approximate the result to zero, the more samples the more reduction of noise, but with the tradeoff of increased latency, which is also an undesired attribute for videogame input.
- Acceleration: By applying an exponential response curve, the parts of the signal with low amplitude are reduced further, including noise. In applications in which linearity is required to achieve consistent results (as is videogame input), this is a undesired tradeoff.
- Sensor fusion: Using a secondary sensor of different nature to discard noise more effectively, each sensor have noise with its own characteristics, but the combination of data from both can help to isolate real input from unintended input.

2.1.2 Scale

Angular rate sensors have a limit in the turn rate they can report on. These are also measured in maximum DPS until they saturate its data channel. For commercially available ARSs this attribute can be configured, usually in predefined steps from 125 DPS to 4000 DPS, referred as ranges or scales.

There are tradeoffs depending in which scale to choose, smaller scales have better granularity (lower DPS per Least Significant Bit), better noise-signal ratio, and smaller zero-rate ratio, but can only report on slower turns. On the contrary, bigger scales have worse granularity, worse noise-signal ratio, an greater zero-rate, but they can report on faster turns.

Since noise and zero-rate in ARSs are not negligible, these tradeoffs can be problematic no matter the chosen scale.

One potential solution is to change the scale dynamically when the current range is too small or too big. But such request to the IMU is relatively slow and a blocking operation, so no reading from the sensor can be taken while the scale is being adjusted. Therefore this method cannot be employed if the system is expecting continuos readings with a smaller interval than the time it takes to change the scale.

2.1.3 Zero-rate

Zero-rate (also known as random walk offset or drift) [6] is a deviation over time of the calibrated zero, doing a pseudo-random walk around the real zero at a much lower frequency than flicker noise (section 2.1.1). In piezoelectric-based IMUs this can be caused by electrical current instability, interference from other electronic components, or subtle changes in temperature.

Zero-rate is measured in maximum DPS of deviation the sensor could reach away from the calibrated zero, and DPS/C$^{\text{o}}$ for changes in temperature. For commercially available ARSs these are in the range of milli-degrees per second, but aggregated over time these can lead to much greater values.

For the application of computer input, this is a big shortcoming of ARSs compared with alternative methods such as optical mouse, which similarly to noise (section 2.1.1), they have inexistent or negligible drift.

Common solutions for zero-rate are:

- Acceleration: By applying an exponential response curve, the parts of the signal with low amplitude are reduced further, including noise. In applications in which linearity is required to achieve consistent results (as is videogame input), this is a undesired tradeoff.
- Multisampling: Contrary to the case of flicker noise, averaging multiple samples **of the same sensor** over time would NOT reduce zero-rate, since zero-rate is random-walk noise, consecutive values will contain very similar deviation, making multisampling useless.
- Self-correction: With software, when a constant angular rate is detected for several consecutive seconds, can be assumed it is not real input and the sensor can be programmatically calibrated.

3 Redundant sensor fusion

3.1 Noise

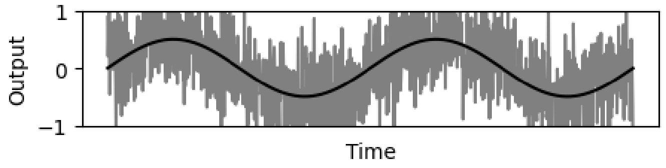
By using multiple physically connected sensors, is possible to take samples in parallel and average their values without increasing latency.

The following figures show a sinusoidal representing actual input, with added simulated noise (normal distribution loc=0 scale=0.5), being processed with different combinations of multisampling and multisensor. Latency being defined as:

$$Latency = \frac{Samples}{Sensors}$$

Figure 1 shows the untreated signal on a single sensor.

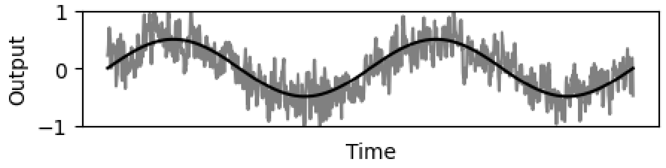
Figure 1: Sensors=1, Samples=1, Latency=1



$$f_1(x) = sin(x) + N(\mu = 0, \sigma = 1)$$

Figure 2 shows result of multisampling method over 4 samples, significantly reducing the noise, but increasing latency.

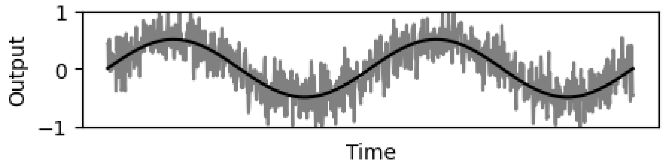
Figure 2: Sensors=1, Samples=4, Latency=4



$$Avg(x,t) = \frac{\sum_{x}^{x+t-1} f_1(x)}{t}$$
$$f_2(x) = Avg(x, 4)$$

Figure 3 shows equivalent noise reduction (using a second sensor signal) but with improved latency. Note that the signal is not just scaled down version of the original noise, but also shows features (relative peaks) of the second sensor averaged signal.

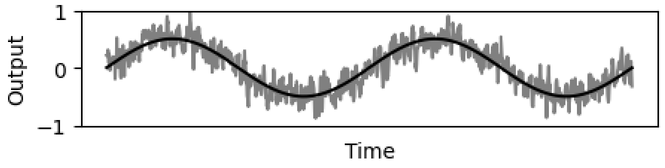
Figure 3: Sensors=2, Samples=4, Latency=2



$$f_3(x) = \frac{Avg(x_1, 2) + Avg(x_2, 2)}{2}$$

Figure 4 shows improved noise reduction by using 2 sensors given the same latency as figure 2.

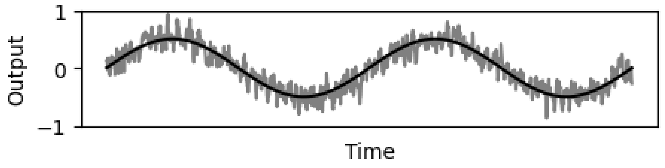
Figure 4: Sensors=2, Samples=8, Latency=4



$$f_4(x) = \frac{Avg(x_1, 4) + Avg(x_2, 4)}{2}$$

Figure 5 shows further improvement of noise reduction by using 3 sensors given the same latency as figure 2.

Figure 5: Sensors=3, Samples=12, Latency=4



$$f_5(x) = \frac{Avg(x_1, 4) + Avg(x_2, 4) + Avg(x_3, 4)}{3}$$

The conclusion is that redundant sensors (of equal or similar characteristics) can be used to either reduce total noise more effectively within the same latency, or to achieve the same level of noise reduction but with less latency.

The benefits of parallelism regarding latency does scale linearly with number of redundant sensors as $latency/n$ (there are no diminishing returns of using more than 2 sensors); while benefits regarding noise reduction only increase at a $1/n$ rate, so there are diminishing returns when adding more sensors.

3.2 Scale

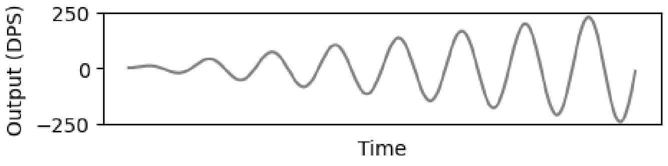
While changing the scale dynamically with a single sensor is problematic (see section 2.1.1), it is possible to obtain the same multi-scale benefits with multiple sensors without these tradeoffs. By using multiple sensors, each one permanently adjusted to a different scale, then using the output of the most suitable sensor (the one with the smaller scale able to fit the input without saturate) a result with lower noise and lower zero-rate is obtained when the input is small, but without limitations in range when the input is greater.

Given a system with multiple sensors S_n, each sensor with a range R_n, and sorted from greater to smaller range, the final output can be defined as:

$$Output = \begin{cases} if(S_1 > R_2) \rightarrow S_1/R_1 \\ else\,if(S_2 > R_3) \rightarrow S_2/R_2 \\ ... \\ else\,if(S_n > R_{n+1}) \rightarrow S_n/R_n \\ ... \\ else(S_n/R_n) \end{cases}$$

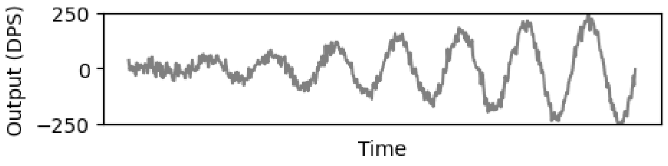
In the following figures an increasing $sin(x)$ function is representing actual input, then noise is used to visualize the undesired characteristics which are relative to the used scale (noise, zero-rate) in the values for the sensors A and B

Figure 6: Actual input



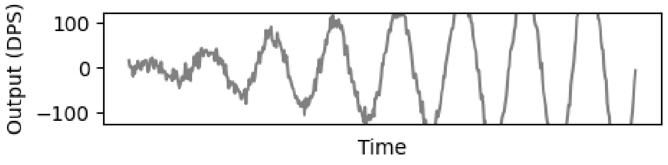
Sensor A is able to fit the full amplitude of the input, but noise is greater than in sensor B.

Figure 7: Sensor A, scale=250



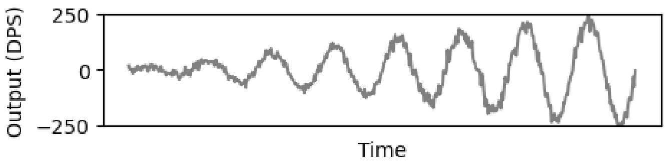
Sensor B noise is relatively smaller, but the input signal makes it to saturate after a certain point.

Figure 8: Sensor B, scale=125



Using the method described above the last figure shows how values of B (with less noise) are used while the signal does not saturate it. But when sensor B starts saturating it switches to the values of sensor A, which is able to capture the full amplitude of the real input.

Figure 9: Dynamic switch sensors A and B



The final result therefore retains the best characteristics of both sensors, range of sensor A with noise-ratio of sensor B (while possible).

3.3 Zero-rate

3.3.1 Basis for calculations

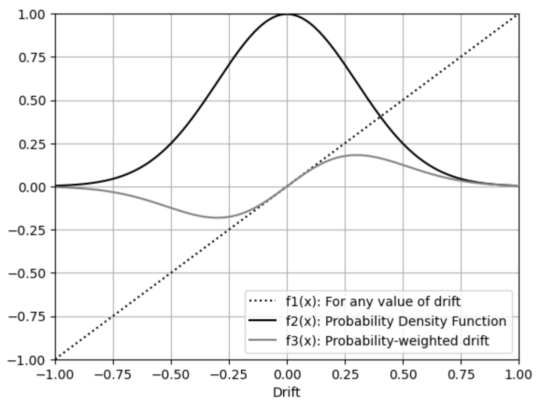
- The mean value of drift will be calculated by repeatedly integrating the expected drift on multiple axis and with multiple sensors, each of the axis and sensors adding additional variables to be integrated.
- To obtain realistic means, normal distribution will be factored in the calculations (see 3.3.2).
- The interval of the definite integrals is -1 to 1, since the negative part is important for making calculations in which opposing values cancel each other.
- To normalize the definite integration result from the (-1 to 1) interval to the (0 to 1) interval, the result must be divided by 2^N, in which N is the number of needed integrations going into the negative side.
- The calculated zero-rate is a value from 0 to 1. The drift is a vector (in 1D, 2D or 3D) but only its length is relevant and the direction is irrelevant. That way the final integrated mean can be a simple scalar above zero.
- It is assumed that the zero-rate is many orders of magnitude smaller than the maximum detection range of the sensor, and therefore it will never reach its limits.

3.3.2 Weighed probability

Taking (f_1) "for any possible value of drift" and multiplying it for the adjusted normal distribution Probability Density Function (f_2), the resulting (f_3) provides "probability weighted drift" values allowing to perform calculations (and integrations) on it as if they were actual sensor output, but also being able to obtain a realistic mean drift scalar (as perceived by the user), which will be the foundation for doing comparisons when adding more axis and more sensors.

For the following figures, the arbitrary values for PDF sigma=0.3 and scale=0.75 will used to simulate a sensor with real world characteristics, but additionally a table with the final results for different PDF values will be shown too.

Figure 10: Probability-weighted drift

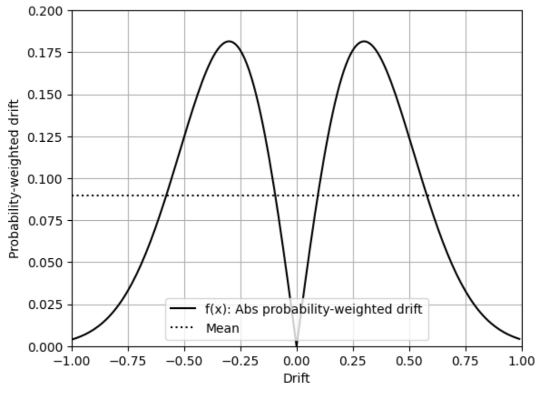


$$f_1(x) = x$$
$$f_2(x) = W(x) = pdf(x, \mu = 0, \sigma = 0.3) * 0.75$$
$$f_3(x) = f_1(x) * f_2(x)$$

3.3.3 One axis

In figure 11, for a single axis and a single sensor, the mean can be integrated by using the absolute drift values.

Figure 11: Axis=1, Sensors=1



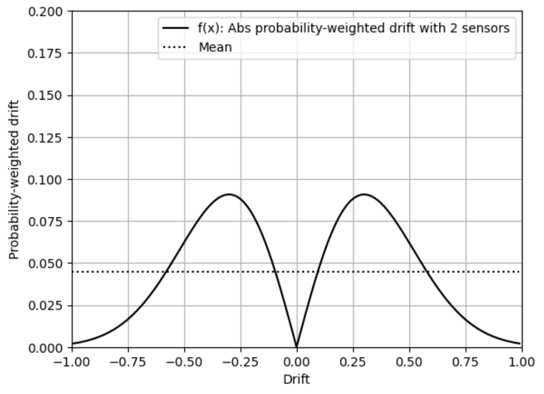
$$f(x) = |W(x)|$$

$$Mean = \frac{1}{2^1}\int_{-1}^{1}|W(x)|dx = \int_{0}^{1}W(x)dx \approx 0.0894$$

In figure 12, by adding a second sensor (also reporting on a single axis), values of both sensors can be averaged as $(x_1 + x_2)/2$, resulting in smaller mean drift.

The negative part (interval $-1, 1$) is used only until the point in which the goal is to start calculating absolute drift. This way an intermediate absolute() transformation between the 1st and 2nd integrations is not required.

Figure 12: Axis=1, Sensors=2



$$f(x) = \left|\int \frac{W(x_1) + W(x_2)}{2}dx_2\right|$$

$$Mean = \frac{1}{2^1}\int_{0}^{1}\int_{-1}^{1}\frac{W(x_1) + W(x_2)}{2}dx_1dx_2 \approx 0.0447$$

So for a single axis, averaging the output of 2 sensors reduces the mean zero-rate to exactly half, independently of the values used to model the probability distribution.

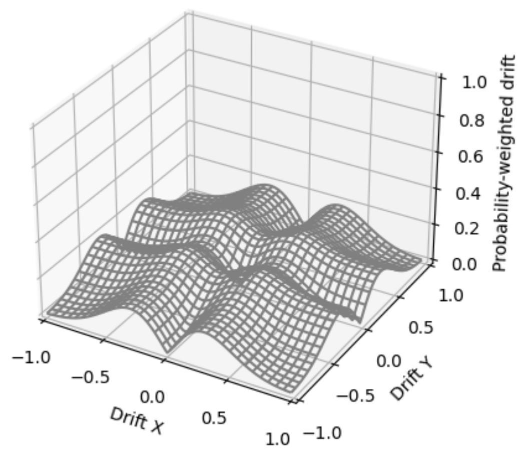
Table 1: Drift reduction on 1 axis

PDF σ	PDF scale	Sens=1	Sens=2	Reduction
0.2	1	0.0798	0.0399	0.5
0.3	**0.75**	**0.0894**	**0.0447**	**0.5**
0.4	0.5	0.0763	0.0381	0.5
0.5	1.25	0.0498	0.0249	0.5

3.3.4 Two axis

In figure 13, when using 2 axis to create a 2-dimensional vector with length $\sqrt{x^2 + y^2}$ and direction $atan2(x/y)$, and use PDF for both axis as described before.

Figure 13: Axis=2, Sensors=1

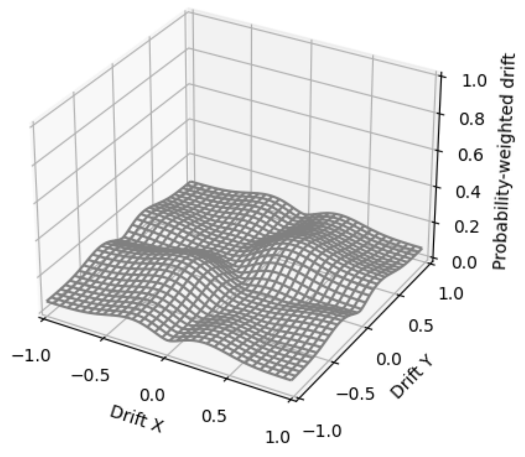


$$f(x) = \sqrt{W(x)^2 + W(y)^2}$$

$$Mean = \frac{1}{2^2}\iint_{-1}^{1}\sqrt{W(x)^2 + W(y)^2}dxdy \approx 0.1408$$

In figure 14, by using an additional sensor the probability-weighted drift is reduced overall, but specially depressed in the quadrants with opposing values (-,+) and (+,-).

Figure 14: Axis=2, Sensors=2



$$U(a, b) = \frac{W(a) + W(b)}{2}$$

$$f(x) = \iint \sqrt{U(x_1, x_2)^2 + U(y_1, y_2)^2}dx_2dy_2$$

$$Mean = \frac{1}{2^4}\iiiint_{-1}^{1}\sqrt{U(x_1, x_2)^2 + U(y_1, y_2)^2}dx_1dy_1dx_2dy_2$$

$$\approx 0.0982$$

So for 2 axis, used as 2-dimensional vectors, averaging the values of a 2nd sensor, it reduces the zero-drift to approximately $\sim 2/3$, varying depending on the characteristics of the sensor (the probability curve),

This solution achieves better results when sigma is higher (when the drift is worse more often), while the scale used for the PDF is irrelevant.

Table 2: Drift reduction on 2 axis

PDF σ	PDF scale	Sens=1	Sens=2	Reduction
0.2	1	0.1366	0.1033	0.7560
0.3	**0.75**	**0.1407**	**0.0981**	**0.6972**
0.4	0.5	0.1137	0.07564	0.6655
0.5	1.25	0.3165	0.2061	0.6511

3.3.5 Conclusion

Redundant sensors can be used effectively to reduce zero-rate in ARS by exploiting the fact that for a set of given random vectors (with signed coordinates) their average is usually closer to the origin or coordinates than any of the individual vectors, or in the worst cases better than at least one of them.

There are diminishing returns both when working with more axis (1D, 2D, 3D), and when adding more sensors.

The same method could be also used with accelerometers to reduce zero-g, or with any other kind of sensor with similar random-walk bias characteristics.

4 Methodology

4.1 Calculations

The claims made in the solution section are backed by theoretical math, such calculations are formal explanations for the results obtained with real-life experimentation, and each result has been always cross-checked with other methods and tools.

The main tool for calculations was NumPy, while also using Scipy Nquad or custom Montecarlo simulations to verify results.

4.2 About figures

Figures for noise (1, 2, 3, 4, 5): Created using Numpy **np.random.normal** for emulating noise, and **np.random.seed** to operate over the same signals repeatedly. With length of $(\pi * 4)$ and resolution of (0.01). Plotted with Matplotlib.

Figures for scale (6, 7, 8, 9): Created using Numpy **np.random.normal** for emulating noise, and **np.random.seed** to operate over the same signals repeatedly. With length of $(\pi * 16)$ and resolution of (0.1). Plotted with Matplotlib.

Figures for zero-rate (11, 12, 13, 14): Created using Numpy and SciPy **scipy.stats.norm.pdf** to calculate Probability Density Distribution. Used **np.meshgrid** and **np.sum** for integrations. With Resolution of (0.01). Plotted with Matplotlib and wireframe for 3D.

4.3 Experimentation

The hypothesis about how to employ redundant sensors were originally tested with breadboard prototypes using a Raspberry Pi Pico and several pairs of IMUs such as 2x *ST LSM6DS33*, 2x *TDK MPU-6050*, and 2x *Bosch BMX055*.

Using these handmade devices as a computer **HID** input allowed to verify such hypothesis with ease, by emulating a mouse and having a real-time and interactive response to the sensors output.

5 Potential applications

In the consumer electronics market, given the increased cost and diminishing returns in performance of using redundant IMUs, its application is possibly relegated to use-cases in which accuracy is a key factor, and enthusiasts are willing to cover the additional costs, such as videogame peripherals for e-sports, or high-performing/racing drones.

In the medical field, redundant lower-grade sensors could be employed when industrial-grade sensors are too big or expensive, as for example electronic prosthetics.

Even though this research is focused in lower-grade IMUs and computer input, the same methods could be used to combine data from multiple industrial-grade sensors and obtain extremely accurate data for scientific or industrial purposes.

References

[1] Merrill, Walter C. "Sensor failure detection for jet engines using analytical redundancy". Journal of Guidance, Control, and Dynamics. (1985). `https://doi.org/10.2514/3.20041`

[2] Mohamed, A., Schwarz, K. "Adaptive Kalman Filtering for INS/GPS". Journal of Geodesy 73. (1999). `https://doi.org/10.1007/s001900050236`

[3] Tadigadapa, S. A. K. M., and K. Mateti. "Piezoelectric MEMS sensors: state-of-the-art and perspectives" Measurement Science and technology. (2009). `https://doi.org/10.1088/0957-0233/20/9/092001`

[4] Bonfield, D. G. "The dynamically tuned gyroscope-A sensor for low cost attitude reference and navigation system". Applications of Modern Gyro Technology. (1977). `https://ui.adsabs.harvard.edu/abs/1977amgt.proc.....B`

[5] Ng, Tuck Wah. "The optical mouse as a two-dimensional displacement sensor". Sensors and Actuators A: Physical 107.1. (2003). `https://doi.org/10.1016/S0924-4247(03)00256-5`

[6] Hiller, Tobias, et al. "Origins and mechanisms of bias instability noise in a three-axis mode-matched MEMS gyroscope". Journal of Microelectromechanical Systems 28.4. (2019). `https://doi.org/10.1109/JMEMS.2019.2921607`

Special thanks to

Guðni Ólafsson.
Michael Przybilski.