

ORIGINAL

## AfKh-OpenIMU Generator: An open-source platform for simulating INS grades and generating datasets for machine learning applications

### AfKh-OpenIMU Generator: una plataforma de código abierto para simular grados de INS y generar conjuntos de datos para aplicaciones de aprendizaje automático

Mohammed Aftatah<sup>1</sup>  , Abdelhak Khalil<sup>2</sup>  , Khalil Zebbara<sup>1</sup>  

<sup>1</sup>IMISR Laboratory, Applied Sciences Faculty Ait Melloul. Agadir, Morocco.

<sup>2</sup>LERSEM Laboratory, Chouaib Doukkali University. El Jadida, Morocco.

**Citar como:** Aftatah M, Khalil A, Zebbara K. AfKh-OpenIMU Generator: An open-source platform for simulating INS grades and generating datasets for machine learning applications. Data and Metadata. 2025; 4:1150. <https://doi.org/10.56294/dm20251150>


**Enviado:** 13-11-2024

**Revisado:** 02-04-2025

**Aceptado:** 18-08-2025

**Publicado:** 19-08-2025

**Editor:** Dr. Adrián Alejandro Vitón Castillo 

**Corresponding Author:** Mohammed Aftatah 

#### ABSTRACT

This paper presents AfKh-OpenIMU Generator, an open-source platform developed in Java and deployed via Docker, aimed at supporting machine learning research in inertial navigation. The objective is to address two major challenges: the high cost of inertial navigation systems (INS) and the lack of large, labeled datasets required for training neural networks. Our platform simulates all six inertial sensors, including three accelerometers and three gyroscopes, using configurable error models that incorporate bias, scale factor, and stochastic noise. From user-defined reference trajectories, it generates raw inertial data and supports large-scale data augmentation by varying noise profiles, enabling the creation of diverse datasets without requiring physical hardware. Simulation results demonstrate high fidelity with real-world INS performance. The generated data yielded root mean square error (RMSE) values of 32,98 meters for low-cost INS, 8,99 meters for industrial-grade INS, and 1,07 meters for tactical-grade INS. In addition, the data augmentation mechanism allows dataset expansion by up to 10 000 times, significantly enhancing training robustness and helping to prevent overfitting in deep learning models. Our platform provides a flexible, low-cost, and reproducible solution for generating realistic inertial data. It facilitates the development and evaluation of machine learning algorithms for sensor fusion and secure navigation, making it particularly valuable for research in GPS-denied environments.

**Keywords:** Inertial Navigation System; Sensor Simulation; Machine Learning Algorithm; Imu Error Modeling; Data Augmentation; Overfitting.

#### RESUMEN

Este artículo presenta AfKh-OpenIMU Generator, una plataforma de código abierto desarrollada en Java y desplegada mediante Docker, diseñada para apoyar la investigación en aprendizaje automático aplicada a la navegación inercial. El objetivo es abordar dos desafíos principales: el alto costo de los sistemas de navegación inercial (INS) y la escasez de grandes conjuntos de datos etiquetados necesarios para entrenar redes neuronales. Nuestra plataforma simula los seis sensores inerciales, incluidos tres acelerómetros y tres giróscopos, utilizando modelos de error configurables que incorporan sesgo, factor de escala y ruido estocástico. A partir de trayectorias de referencia definidas por el usuario, genera datos inerciales en bruto y admite una amplia ampliación de datos mediante la variación de los perfiles de ruido, lo que permite crear conjuntos de datos diversos sin necesidad de hardware físico. Los resultados de la simulación demuestran una alta fidelidad en comparación con el rendimiento real de los sistemas INS. Los datos generados arrojaron valores de error cuadrático medio (RMSE) de 32,98 metros para INS de bajo costo, 8,99 metros para INS

de grado industrial y 1,07 metros para INS de grado táctico. Además, el mecanismo de aumento de datos permite expandir el conjunto hasta 10 000 veces, lo que mejora significativamente la robustez del entrenamiento y ayuda a prevenir el sobreajuste en modelos de aprendizaje profundo. Nuestra plataforma ofrece una solución flexible, de bajo costo y reproducible para generar datos inerciales realistas. Facilita el desarrollo y la evaluación de algoritmos de aprendizaje automático para fusión sensorial y navegación segura, siendo especialmente útil en entornos sin cobertura GPS.

**Palabras clave:** Sistema de Navegación Inercial; Simulación de Sensores; Algoritmo de Aprendizaje Automático; Modelado de Errores del IMU; Aumento de Datos; Sobreajuste.

## INTRIDUCTION

Modern intelligent systems have become pervasive across critical domains ranging from aviation and robotics to automotive applications and beyond.<sup>(1)</sup> While these technologies drive innovation, their widespread adoption creates an urgent need for secure, continuous, and reliable navigation solutions.<sup>(2,3)</sup> The consequences of navigation failures—whether a catastrophic automotive collision due to positioning errors or a flawed aircraft landing extend beyond endangering lives to imposing severe economic and societal costs. To address these challenges, diverse navigation systems have been developed. Global Positioning Systems (GPS), which rely on satellite networks to deliver precise worldwide positioning data, remain vulnerable to signal disruptions caused by environmental obstructions or intentional attacks such as GPS spoofing and GPS jamming.<sup>(4,5,6)</sup> These two types of attacks, often conducted by malicious actors, pose a serious threat to critical applications that depend on such systems. GPS spoofing involves broadcasting fake signals to deceive the receiver, causing significant trajectory deviations or deliberately guiding the target to a location chosen by the attacker.<sup>(1,7,8,9)</sup> On the other hand, GPS jamming consists of emitting high-power signals on the same frequency as GPS transmissions, effectively overwhelming the legitimate signals and leading the system to compute incorrect positioning information.<sup>(1)</sup>

As a complementary solution, Inertial Navigation Systems (INS) provide autonomous operation through onboard triaxial accelerometers and gyroscopes that measure linear accelerations and angular rates.<sup>(10,11)</sup> Such a system uses mechanical sensors embedded in the vehicle, operating independently of external signals—unlike satellite-based systems like GNSS. This characteristic enables continuous navigation under all conditions, including in environments where satellite signals are unavailable or unreliable. However, INS technologies face two critical limitations: inherent sensor errors that accumulate rapidly during integration, and the high cost of inertial measurement units (IMUs).<sup>(9,12)</sup> Physical IMUs also introduce additional complexities, including configuration challenges and constrained testing scenarios due to hardware limitations.<sup>(13)</sup>

To overcome the limitations associated with each individual navigation system, numerous solutions have been proposed in the literature, exploiting the complementarities between various sensors. Among these, GPS/INS fusion using the Kalman filter remains one of the most widely adopted approaches. It operates through a two-step process: a prediction step that estimates the system's state based on a dynamic model, followed by a correction step using a measurement model. The Kalman filter also allows for the integration of data from other sensors beyond GPS and INS, such as odometry systems, LiDAR, or cameras.

However, the original Kalman filter struggles to handle nonlinear systems. To address this limitation, several extensions have been developed, including the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). Yet, these enhanced versions often come with a major drawback: significantly increased computational complexity.

With the rise of parallel computing enabled by GPUs – replacing traditional CPU processing – and the rapid advancement of machine learning algorithms, new alternatives have emerged. Machine learning has found its place in navigation due to its ability to model complex and nonlinear systems, a characteristic inherent to the behavior and error models of inertial sensors. In fact, the error modeling and integration equations of such sensors are nonlinear in nature, making machine learning a strong candidate for solving this type of problem.

Nevertheless, despite the availability of GPU acceleration, supervised learning algorithms still require large volumes of data for effective training. Many existing works rely on real INS datasets, which are often limited in size and diversity, leading to potential overfitting during model training.

Overfitting is a major phenomenon closely associated with machine learning models, particularly deep learning algorithms. Models such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory networks (LSTMs) are widely used in navigation applications due to their ability to capture complex temporal and spatial patterns. However, training these models comes with a significant challenge: overfitting. One of the primary causes of overfitting in deep learning is the use of a limited dataset during training. When insufficient data is available, the model tends to memorize the training examples rather

than learning to generalize to unseen data. As a result, while the model may perform well on the training set, its performance on new or real-world data degrades significantly. Synthetic data generation and augmentation are increasingly used to address this limitation and to support robust training in machine learning frameworks.

Rare research discusses the generation of INS datasets for free use; the authors of propose MAGF-ID,<sup>(14)</sup> a valuable experimental dataset for MIMU and GFINS research, featuring 54 real inertial sensors and ground-truth trajectories. However, its limitations include fixed, hardware-dependent noise profiles, constrained scenario diversity, and a non-scalable data volume. While this UAV dataset provides valuable real-world visual-inertial data with RTK ground truth across 50-500m altitudes - addressing a notable gap in low-altitude flight scenarios - it shares common limitations of experimental datasets: the fixed 28 sequences cannot cover all potential flight conditions, sensor noise characteristics are hardware-dependent and unmodifiable,<sup>(15)</sup> and the dataset size cannot be expanded without new costly field campaigns. This paper presents a unique outdoor aerial visual-inertial-LiDAR dataset captured using a multi-sensor payload to promote GNSS-denied navigation research.<sup>(16)</sup> The experiment consists of hardware-synchronized monocular images, IMU measurements, 3D LiDAR point clouds, and high-precision RTK-GNSS-based ground truth. This dataset was designed to facilitate the development of visual-inertial-LiDAR odometry and mapping algorithms, visual-inertial navigation algorithms, object detection, segmentation, and landing zone detection algorithms based on real-world drone and full-scale helicopter data. However, despite its richness, the dataset faces notable limitations: the high cost of the multi-sensor payload, the complexity of the calibration and synchronization procedures, and the operational constraints (requiring large-scale vehicles like drones and helicopters).

This paper presents a novel simulation platform designed to overcome key barriers in navigation research. It accurately models all six IMU sensors—three accelerometers and three gyroscopes—using configurable error profiles that replicate commercial, tactical, and industrial-grade devices. The platform enables the generation of scalable synthetic datasets tailored for training deep neural networks aimed at low-cost inertial error correction, using credible tactical-grade INS data as a reference. Developed as an open-source framework in Java and running in Docker to avoid compatibility with the operating system or the conflict with other applications, the platform proposes a user graphical interface for parameter selection, including IMU grade, reference trajectory upload, and dataset scaling. A MATLAB backend handles the simulation of the inertial sensors to then generate the raw data. Hosted on GitHub, the platform eliminates both financial and logistical barriers, supporting accessible and reproducible research in advanced and resilient navigation system design. It is freely available at the following link: <https://github.com/GenerationINSDataset/AfKh-OpenIMU>.

The remainder of this paper is organized as follows: section 2 details the proposed method, including INS modeling and its principal inherent errors, the Java user interface with Docker deployment, and the experimental configuration. Section 3 validates the proposed approach and demonstrates the feasibility of our developments through results and discussion and presents some directions for future work. Finally, section 4 concludes the paper.

**Table 1.** Comparison of rare existing research on dataset generation for simulating inertial navigation systems and our proposed solution

Authors	Year	Model	Results	Limitations
Yampolsky et al. <sup>(14)</sup>	2024	MAGF-ID	Provides experimental dataset with 54 real sensors and ground-truth trajectories	Fixed hardware-dependent noise, limited diversity, non-scalable data volume
Lyu et al. <sup>(15)</sup>	2024	UAV dataset	Rich multi-sensor data for UAVs and helicopters	Sensor noise characteristics are hardware-dependent and unmodifiable
Thalagala et al. <sup>(16)</sup>	2024	MUN-FRL	IMU measurements, 3D LiDAR point clouds, and high-precision RTK-GNSS-based	High cost, complex synchronization, requires large-scale vehicles
Aftatah et al.	2025	AfKh-OpenIMU Generator	Generates scalable synthetic datasets with configurable IMU error models; includes GUI and Docker deployment	-----

## METHOD

We conducted experiments to validate our proposed method, with the aim of designing an open-source simulation platform capable of generating large-scale augmented inertial navigation datasets. The proposed method begins with the accurate mathematical modeling of the six inertial sensors. This includes the incorporation of key sensor error sources such as bias, scale factor, and noise. The theoretical models are then translated into executable simulation code within the MATLAB environment to generate realistic inertial data. In the second stage, a user-friendly graphical interface is developed using Java, allowing users to define

input parameters such as sensor grade, reference trajectory, and dataset scaling. This interface is deployed as a Docker container image to ensure platform independence and ease of integration. In order to visualize the estimated trajectory produced by each INS grade, the mechanization process is also implemented within the Docker image. Mechanization refers to the set of navigation equations that transform raw sensor data, linear accelerations and angular velocities, into exploitable navigation information such as three-dimensional position and orientation (attitude). The attitude is described using the Euler angles in the following order: roll, pitch, and yaw. Specifically, roll represents the rotation about the x-axis, pitch is the rotation about the y-axis, and yaw is the rotation about the z-axis. The following subsections describe each component of the method in detail.

### Inertial sensor modeling

An inertial navigation system (INS) is composed of two essential components: an inertial measurement unit (IMU) and a processing unit.<sup>(17)</sup> The IMU integrates three gyroscopes and three accelerometers to measure movements along three axes, delivering raw sensor data.<sup>(18)</sup> The processing unit then uses these measurements, based on the mechanization equations, to compute key navigation parameters, including attitude, velocity, and position.<sup>(19)</sup> In this work, the simulated IMU setup consists of three accelerometers and three gyroscopes. The simulation process involves calculating accelerations and angular velocities based on 3D positional data along the X, Y, and Z axes. The 3D trajectory used is a real trajectory that emulates a vehicle's movement. For IMU modeling, the East-North-Up (ENU) reference frame is adopted to express the equations. Throughout this study, the ENU frame, also referred to as the navigation frame (n-frame), serves as the primary reference.<sup>(20)</sup> The body frame (b-frame) is located at the center of the IMU.<sup>(21)</sup> The IMU modeling equations in the n-frame are presented in equations (1), (2), and (3).<sup>(13)</sup>

$$[\varphi \quad \theta \quad \psi] = \begin{bmatrix} \frac{\partial \left( \arctan \frac{\partial y / \partial t}{\partial x / \partial t} \right)}{\partial t} & \frac{\partial \left( \arctan \frac{\partial z / \partial t}{\partial x / \partial t} \right)}{\partial t} & \frac{\partial \left( \arctan \frac{\partial z / \partial t}{\partial y / \partial t} \right)}{\partial t} \end{bmatrix}^n \quad (1)$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix}^b = \begin{bmatrix} 1 & \frac{\sin \varphi \sin \theta}{\cos \theta} & \frac{\cos \varphi \sin \theta}{\cos \theta} \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \frac{\sin \varphi}{\cos \theta} & \frac{\cos \varphi}{\cos \theta} \end{bmatrix} * \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}^b = \frac{\partial}{\partial t} \begin{bmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial t} \end{bmatrix}^n - C_n^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} * C_n^b \begin{bmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial t} \end{bmatrix}^n \quad (3)$$

The components of the three equations presented above are detailed as follows:  $[\varphi, \theta, \psi]$  represent the Euler angles, which describe the orientation of the body frame relative to the navigation frame.  $[x, y, z]$  denote the 3D position coordinates expressed in the East-North-Up frame. The variable  $t$  corresponds to time.  $[p, q, r]$  are the components of the angular velocity vector, representing rotational rates around the body frame axes.  $[(\varphi), (\theta), (\psi)]$  are the time derivatives of the Euler angles.  $[f_x, f_y, f_z]$  correspond to the linear accelerations measured along the three axes of the body frame.  $C_n^b$  is the transformation matrix from the navigation frame to the body frame enables the conversion of vectors between these frames, while  $C_b^n$  a separate conversion matrix is used to map quantities from the body frame to the ENU frame.

### Sensor errors modeling

Inertial navigation systems, relying on mechanical sensors, are inherently subject to various sources of error that significantly degrade their precision over time. In this study, we focus on three major error sources, as they have the most critical impact on the drift behavior of such systems. Specifically, bias is modeled as

a constant offset added to the true sensor output, typically caused by imperfections in manufacturing or sensor calibration.<sup>(22,23)</sup> The scale factor error is represented as a multiplicative deviation from the reference value, often arising from inaccuracies in the sensor's sensitivity or response characteristics.<sup>(22,23)</sup> Finally, noise is modeled as random additive fluctuations, mainly introduced by electronic disturbances and internal thermal variations. By concentrating on these dominant error sources, the model effectively captures the key factors influencing INS performance. The modeling of the previously described errors is detailed in the following equations.<sup>(13)</sup>

$$\begin{cases} a_{measured} = (1 + SF_{accelerometer}) * a_{true} + b_{accelerometer} + \eta_{accelerometer} \\ \omega_{measured} = (1 + SF_{gyroscope}) * \omega_{true} + b_{gyroscope} + \eta_{gyroscope} \end{cases} \quad (4)$$

(5)

In the presented modeling,  $\omega_{measured}$  represents the measured angular rate, while  $\omega_{true}$  denotes the true angular rate. The term  $b_{gyroscope}$  corresponds to the gyroscope bias,  $\eta_{gyroscope}$  to the gyroscope noise, and  $SF_{gyroscope}$  to the gyroscope scale factor. Similarly,  $a_{measured}$  refers to the measured linear acceleration,  $a_{true}$  to the true linear acceleration,  $b_{accelerometer}$  to the accelerometer bias,  $\eta_{accelerometer}$  to the accelerometer noise, and  $SF_{accelerometer}$  to the accelerometer scale factor.

Figure 1 illustrates the IMU modeling process within the body frame, offering a visual overview of the simulation approach adopted in this study. The developed platform provides the flexibility to modify error parameters, enabling the increase or decrease of their values to simulate various scenarios and evaluate their effects on system performance.

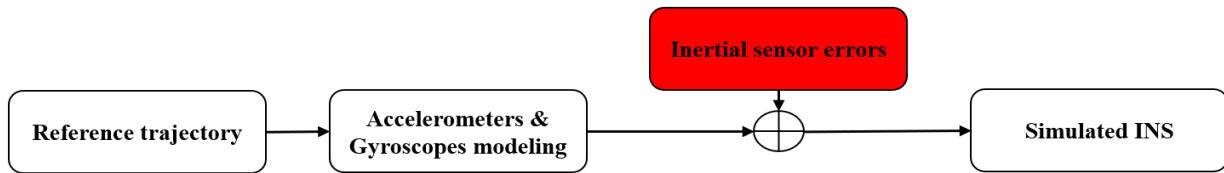
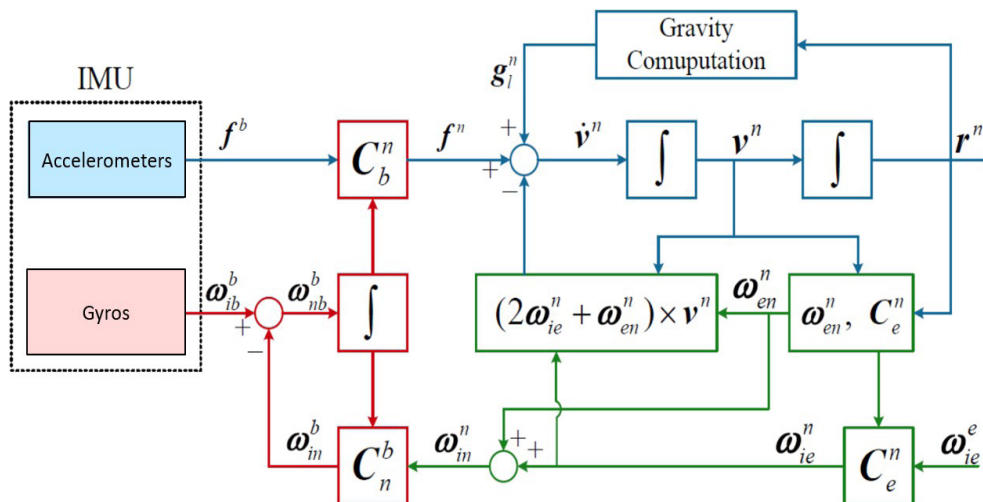


Figure 1. Methodology adopted for IMU modeling within the body frame<sup>(24)</sup>

### Inertial sensors mechanization

Once the six degrees-of-freedom inertial sensors—comprising three accelerometers and three gyroscopes—are mathematically modeled, they produce raw inertial measurements in the form of specific forces and angular velocities. At this stage, two complementary processes are initiated. The first is the mechanization, which converts the raw sensor data into navigable information. The second is data augmentation, which artificially expands the dataset by varying sensor noise parameters to improve the robustness and generalization of learning algorithms. The focus here is on the mechanization process, which is a fundamental step in any inertial navigation system.





Inertial sensor mechanization is defined as the process of computing position, velocity, and orientation over time from raw inertial data using the laws of motion. This is achieved through the continuous integration of a set of navigation equations within a defined reference frame, typically the local-level frame known as North-East-Down (NED). The mechanization process begins by updating the orientation using angular velocity data, which determines the system’s attitude in terms of roll, pitch, and yaw. Roll describes the rotation around the x-axis, pitch around the y-axis, and yaw around the z-axis. The next step involves computing the velocity by integrating the specific force measurements after compensating for gravity, followed by position estimation through the integration of velocity. This process provides a complete navigation solution and enables real-time visualization of the estimated trajectory for each sensor grade. Figure 2 details this mechanization process as implemented in the navigation frame.

Java user interface and Docker deployment

To facilitate the use of the inertial system simulation, we developed a platform based on two frameworks: React.js for the frontend and Spring Boot for the backend. This platform will later be packaged as a Docker container image to simplify deployment and usage across different environments.<sup>(26)</sup> Through the user interface, users can upload a reference trajectory defined in cartesian coordinates and specify their requirements regarding the grade of the inertial system to be simulated. They can also set the sampling time and define a data augmentation factor, which controls how many additional data points are generated from the original reference trajectory. Following this, the user can generate a text file containing the ground truth and all the necessary parameters. This file can then be easily injected into the MATLAB code to finally generate the full dataset. Figure 3 shows a screenshot of the graphical interface of our platform.

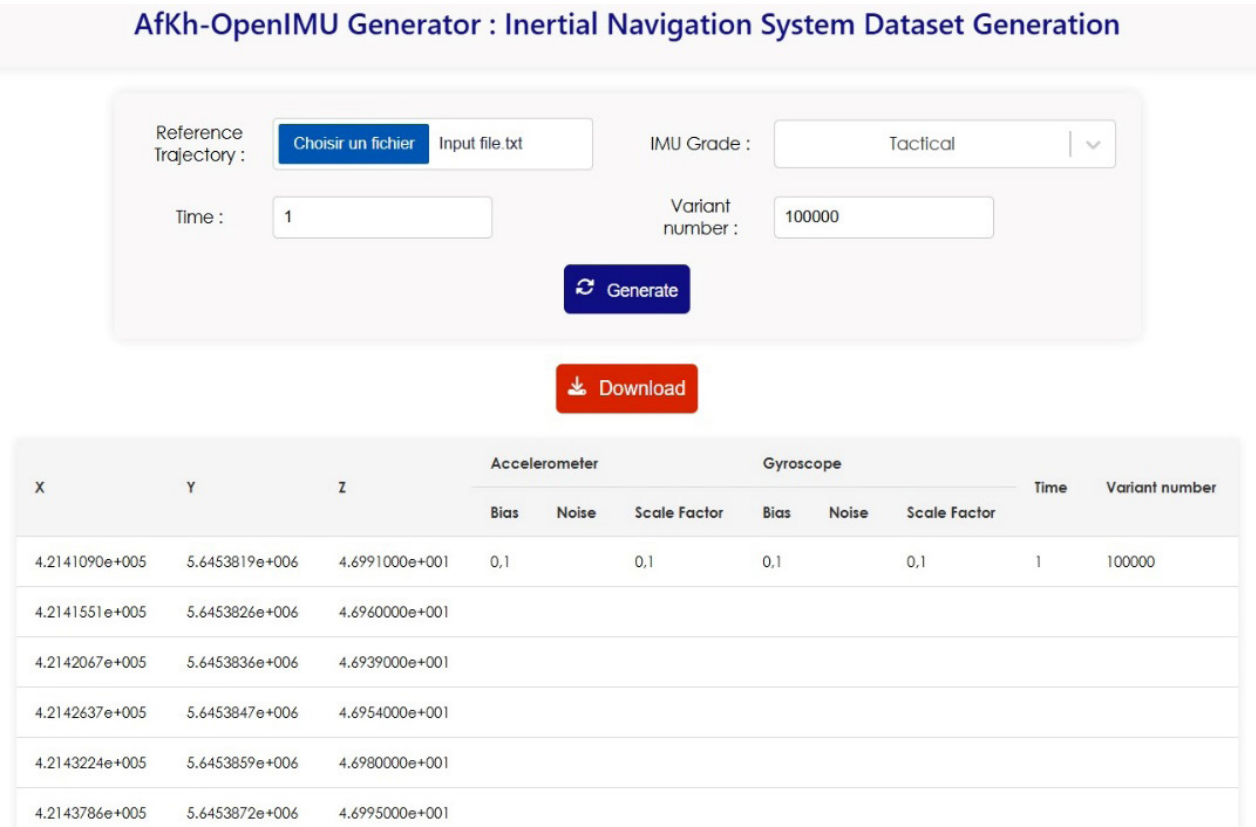


Figure 3. Interface for reference trajectory input and simulation configuration

The developed platform includes the complete source code for its components, combining a user-friendly graphical interface built with React.js and Spring Boot, along with MATLAB scripts for simulating three grades of Inertial Navigation Systems (INS): tactical, industrial-grade, and low-cost. Additionally, a MATLAB-based data generator is integrated, responsible for augmenting the dataset by varying key error parameters such as scale factor, bias, and stochastic noise. All these components are containerized into a single Docker image. The pivotal role of this containerization lies in ensuring seamless deployment across various environments, regardless of operating system compatibility or language dependencies. This functionality significantly enhances the platform’s usability and reproducibility, making it feasible to execute and scale the system across a wide range of heterogeneous computing environments.

### Simulation process

Our proposed method, illustrated in the diagram shown in figure 4, consists of four essential steps. In Step 1, the user (researcher) defines their requirements through a simple and dedicated graphical interface by uploading the reference trajectory, selecting the inertial system grade, setting the data multiplier and time, and then clicking “Generate”. Step 2 produces a text file that contains all the specified parameters and user inputs. In Step 3, this text file is passed to a MATLAB-coded main program that simulates the inertial navigation system and generates the corresponding dataset. Finally, Step 4 delivers the output: a “.mat” file dataset for training, a graphical comparison between the reference and estimated trajectories, and the simulated inertial data in terms of accelerations and angular rates.

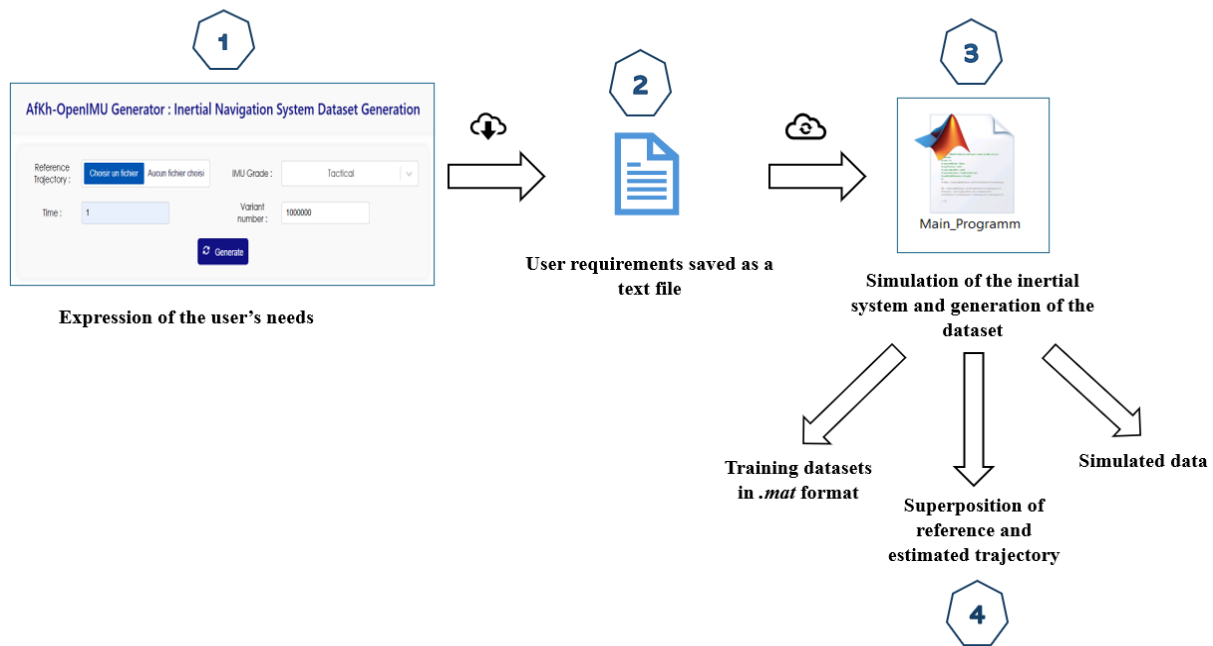


Figure 4. Overview of the proposed four-step data generation process

### AfKh-OpenIMU Generator

The integration of these components, including inertial sensor modeling, error injection, mechanization, user interface development, Docker deployment, and the simulation workflow, forms the architecture of our developed platform called AfKh-OpenIMU Generator. This platform is particularly beneficial when working with machine learning algorithms, as it enables the generation of sufficiently large and diverse datasets to mitigate overfitting problems. The overall diagram illustrating this solution is presented in figure 5.

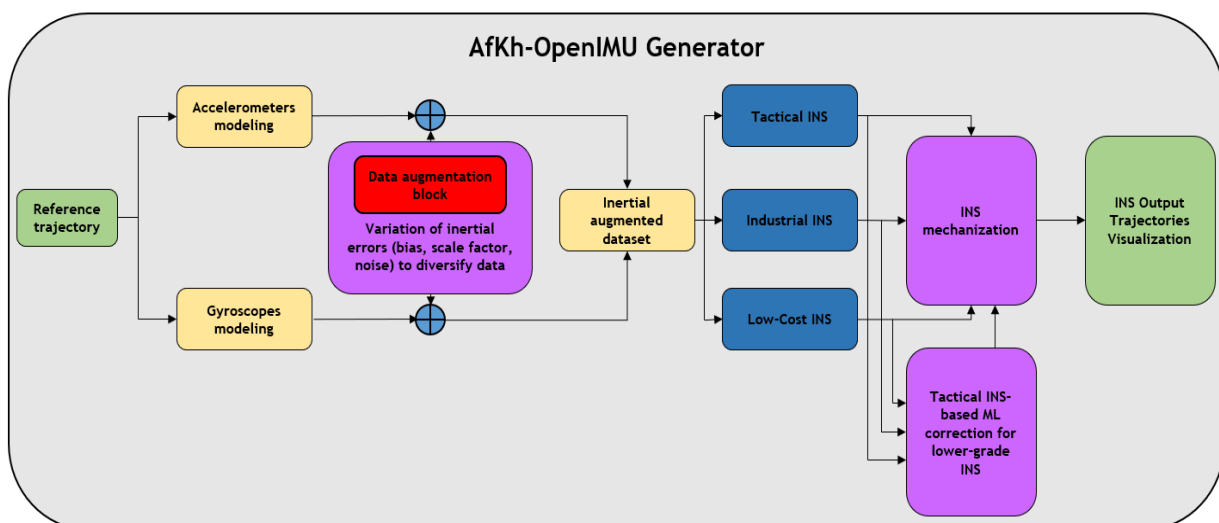


Figure 5. Architectural flowchart of the proposed AfKh-OpenIMU Generator

### Experimental configuration

In this study, we employed a real-world trajectory defined by positions in a navigation frame, with coordinates represented in a 3D Cartesian system. The reference trajectory was obtained from Marrakech City, Morocco, emulating a realistic urban transportation scenario. The trajectory comprised 620 discrete points and has a total length of 1653 meters, capturing the dynamics of vehicular motion. The reference path was initially generated using Google Earth Pro and exported as a Keyhole Markup Language (KML) file. While KML is suitable for visualization, its binary structure hinders direct data processing. To enable data manipulation, the KML file was converted into a plaintext format (comma-delimited TXT file) using GPS Visualizer—an online tool for geodata transformation. The output file contained coordinates in WGS84 (decimal degrees) with columns for latitude, longitude, and altitude. For practical navigation and analysis, the coordinates were converted from WGS84 (geodetic degrees) to WGS84 (UTM) via the World Coordinate Converter (WCC) online platform. This step projected the curvilinear geographic coordinates into a local Cartesian East-North-Up (ENU) frame, simplifying distance and velocity calculations. The reference trajectory used in this study is illustrated in two stages: Figure 6 shows the satellite view of the trajectory extracted from Google Earth. Figure 7 details this same reference projected in the navigation frame and visualized in 2D using MATLAB displays.



Figure 6. Satellite view of reference trajectory in Marrakech city using Google Earth

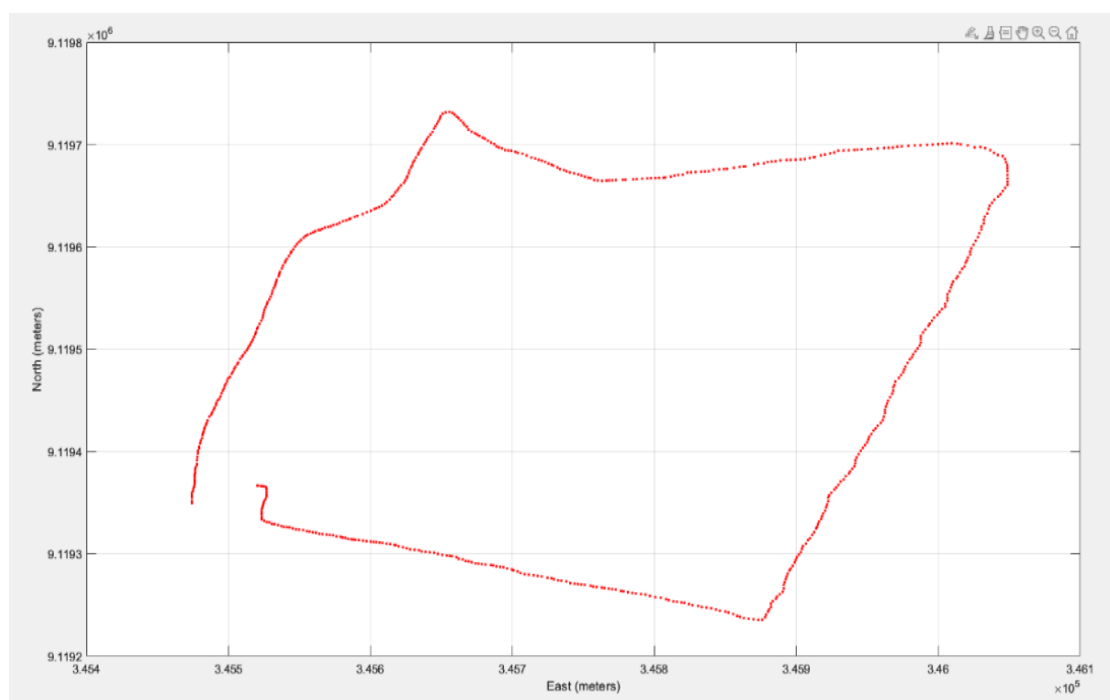


Figure 7. Complete reference trajectory projected in the navigation frame and visualized in 2D using MATLAB



## RESULTS

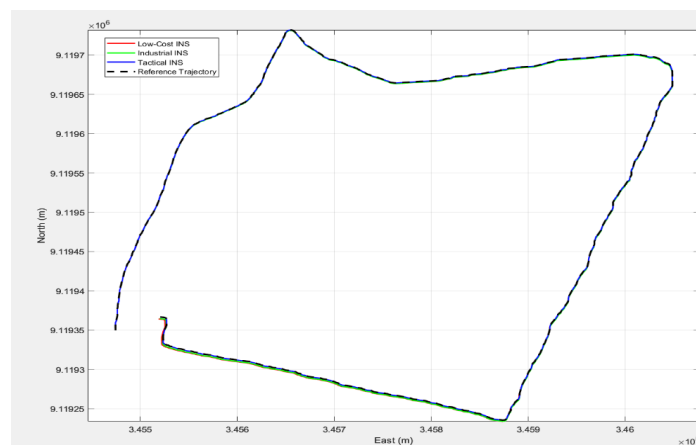
To validate the proposed modeling approach and demonstrate the feasibility of our developments, a series of experiments, detailed in the previous section, were conducted to evaluate the platform's performance. For this evaluation, two key criteria were considered: the comparison of simulated trajectories with reference trajectories, and the calculation of the Root Mean Square Error (RMSE) between them. Additionally, the performance of the developed platform was compared to that of real inertial sensors to assess its realism and accuracy.

### Trajectory comparison

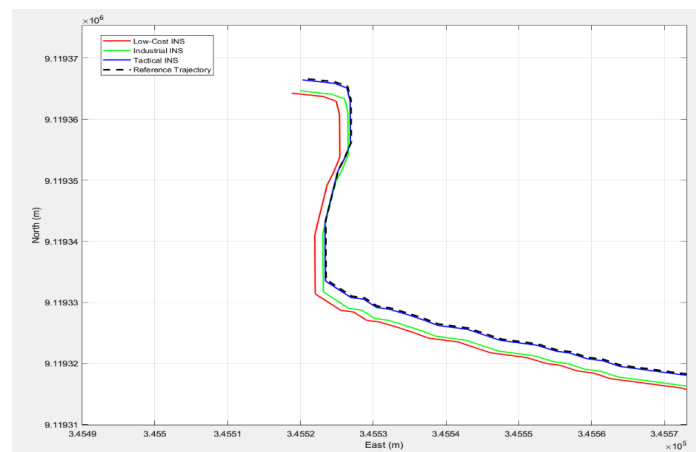
For the comparison of trajectories, four trajectories were superimposed in 2D for better visualization: the ground truth, the estimated trajectory produced by the low-cost INS, the generated trajectory by the industrial-grade INS, and the trajectory obtained from the tactical-grade INS. This superposition is crucial to highlight the performance of our proposed approach. However, in the freely available platform hosted on GitHub, the user must select a single INS grade, and the platform will generate the corresponding data in 3D, not just in 2D, for the selected grade. Figure 8 shows the superposition of the four trajectories.

Figure 8 (a) shows the complete superposition of the reference trajectory and those estimated by the three INS grades, clearly demonstrating a strong consistency with the theoretical expectations. Specifically, the trajectory estimated by the low-cost INS deviates the most from the reference due to its higher error levels. In second position, the industrial-grade INS trajectory exhibits a smaller deviation, reflecting improved accuracy. Finally, the path generated by the tactical-grade INS closely follows the reference trajectory, due to its minimal error characteristics.

Figure 8 (b) presents a zoomed-in view of the final portion of the trajectory to better visualize the accumulation effect of inertial errors over time. This focused view reveals how these errors tend to grow progressively along the trajectory, particularly in the case of the low-cost INS, where error drift is most pronounced. This observation clearly illustrates the need for correcting inertial errors in low-grade systems, reinforcing the motivation behind employing machine learning algorithms for error compensation.



(a)



(b)

**Figure 8.** 2D comparison between ground truth and estimated trajectories for Low-Cost, industrial-grade, and tactical-grade INS (a), with a zoomed overview (b)

## RMSE evaluation

The RMSE (Root Mean Square Error) is a fundamental criterion used to evaluate the performance of the proposed method. In this study, the RMSE is calculated for three simulated grades of INS in the East, North, and Up directions for each grade. The RMSE is computed by comparing the estimated trajectory of each INS grade with the reference trajectory, as depicted in table 2.

INS grade	RMSE (meter)			
	East	North	Up	Total
Low Cost	9,22	16,94	26,75	32,98
Industrial	0,58	1,28	8,88	8,99
Tactical	0,01	0,13	1,06	1,07

The results show that the low-cost INS exhibits the highest RMSE values in all three directions, due to the significant errors inherent to such systems, mainly resulting from the use of low-quality materials and sensors. The industrial-grade INS ranks second, demonstrating a noticeable reduction in RMSE across the East, North, and Up directions, owing to improved manufacturing quality, better calibration processes, and enhanced sensor performance. Finally, the tactical-grade INS shows a significant reduction in RMSE compared to the reference trajectory, achieving high precision in all three directions. This superior performance is mainly attributed to the use of high-grade materials, rigorous calibration, advanced sensor technology, and the implementation of sophisticated error compensation techniques, which are typical features of tactical-grade systems.

Moreover, the obtained results show a close correspondence with the expected performances of real sensors, confirming the effectiveness of our modeling approach for both the sensors and their associated errors, as well as the relevance of the parameter choices made for the three principal error sources affecting each grade.

## Custom Dataset Generation Output

Although the dataset generation process does not produce a visual output suitable for direct inclusion as a figure, the result of this stage is a structured and augmented dataset automatically generated in “.txt” and “.mat” formats. This dataset reflects the user’s selections defined through the graphical interface, including the INS grade, reference trajectory, number of sequences, and error profile configuration. The platform ensures that each generated dataset is customized to the specified simulation needs, supporting varied use cases such as machine learning training, error analysis, and trajectory reconstruction. The flexibility of the system to generate on-demand, diverse datasets represent a key contribution, especially in domains where real-world inertial data acquisition is costly or limited.

## DISCUSSION

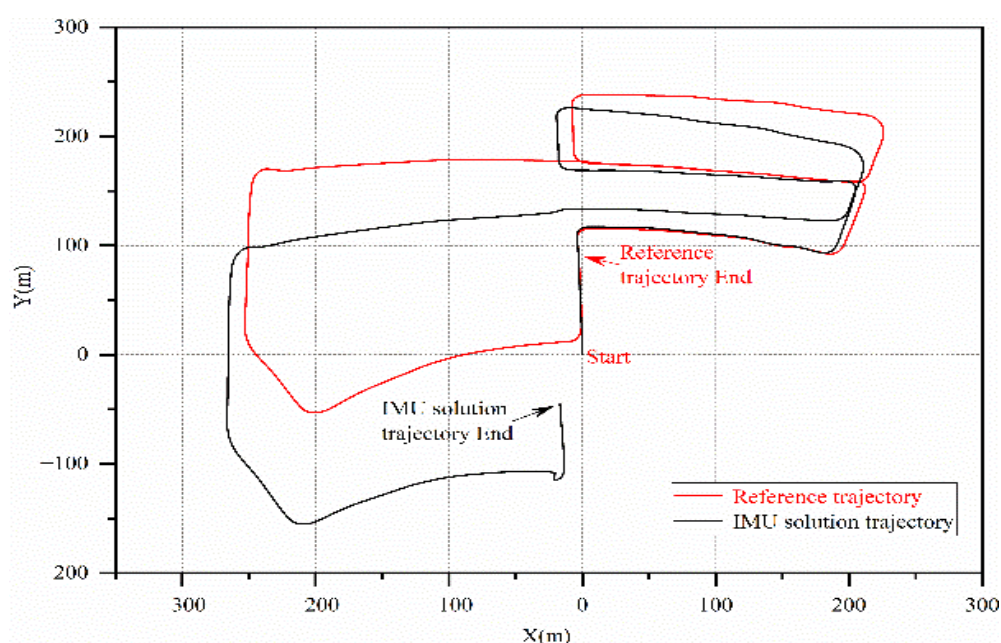


Figure 9. Performance of real Low-Cost INS in a real-world transport scenario<sup>(27)</sup>

Although the developed platform successfully achieves its initial objectives by simulating three grades of inertial navigation systems and supporting large-scale data augmentation to overcome underfitting, several enhancements can be envisioned. One key area for improvement is the simulation of additional sources of sensor errors, particularly those caused by temperature variations and sensor misalignment, which are commonly encountered in real-world hardware. Moreover, future versions of the platform could benefit from a more structured and optimized data management system.

## CONCLUSIONS

This study addresses the growing need for secure, adaptable, and cost-effective navigation solutions in intelligent systems. By introducing a modular and extensible simulation framework, AfKh-OpenIMU Generator, we enable the generation of synthetic inertial datasets at scale, independent of physical hardware constraints. The approach fosters reproducibility, promotes data diversity, and supports the design of robust machine learning models for navigation in both nominal and degraded environments. Beyond its technical implementation, the platform reflects a methodological shift toward virtualized experimentation, facilitating innovation in sensor modeling, data augmentation, and navigation security. It establishes a versatile foundation for future research focused on resilient sensor fusion strategies and AI-driven navigation frameworks under constrained conditions.

## BIBLIOGRAPHIC REFERENCES

1. Aftatah M, Zebbara K. Robust ConvNet-Kalman Filter Integration for Mitigating GPS Jamming and Spoofing Attacks Basing on Inertial Navigation System Data. *Data Metadata*. 2024 <https://doi.org/10.56294/dm2024.405>
2. Panigrahi PK, Bisoy SK. Localization strategies for autonomous mobile robots: A review. *J King Saud Univ Comput Inf Sci*. 2022 Sep;34(8):6019-39. <https://doi.org/10.1016/j.jksuci.2021.02.015>
3. He Y, Li J, Liu J. Research on GNSS INS & GNSS/INS Integrated Navigation Method for Autonomous Vehicles: A Survey. *IEEE Access*. 2023;11:79033-55. <https://doi.org/10.1109/ACCESS.2023.3299290>
4. Aftatah M, Zebbara K. A Comprehensive Survey on Secure Navigation for Intelligent Systems: Artificial Intelligence Approaches to GPS Jamming and Spoofing Detection. In: Mejdoub Y, Elamri A, editors. *Proceedings of the International Conference on Connected Objects and Artificial Intelligence (COCIA2024)*. Cham: Springer Nature Switzerland; 2024. p. 105-10. [https://doi.org/10.1007/978-3-031-70411-6\\_17](https://doi.org/10.1007/978-3-031-70411-6_17)
5. Omali TU, Akpata SBM. Global Positioning System Technology: Theory and the Methods of its Application. *Int J Sci Res Multidiscip Stud*. 2024 May;10(5):70-6.
6. Chefrour D. Evolution of network time synchronization towards nanoseconds accuracy: A survey. *Comput Commun*. 2022 Jul;191:26-35. <https://doi.org/10.1016/j.comcom.2022.04.023>
7. Sathaye H, Lenders V. An Experimental Study of GPS Spoofing and Takeover Attacks on UAVs.
8. Ghanbarzade A, Soleimani H. GNSS/GPS Spoofing and Jamming Identification Using Machine Learning and Deep Learning. *arXiv [Preprint]*. 2025 Jan 4 [cited 2025 Aug 7]; Available from: <https://doi.org/10.48550/arXiv.2501.02352>
9. Jung JH, Hong MY, Choi H, Yoon JW. An Analysis of GPS Spoofing Attack and Efficient Approach to Spoofing Detection in PX4. *IEEE Access*. 2024;12:46668-77. <https://doi.org/10.1109/ACCESS.2024.3382543>
10. Zhuang Y, et al. Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches. *Inf Fusion*. 2023 Jul;95:62-90. <https://doi.org/10.1016/j.inffus.2023.01.025>
11. Jalal A, Quaid MAK, Tahir SB ud din, Kim K. A Study of Accelerometer and Gyroscope Measurements in Physical Life-Log Activities Detection Systems. *Sensors*. 2020 Jan;20(22):6670. <https://doi.org/10.3390/s20226670>
12. Zheng T, Xu A, Xu X, Liu M. Modeling and Compensation of Inertial Sensor Errors in Measurement Systems. *Electronics*. 2023 Jan;12(11):2458. <https://doi.org/10.3390/electronics12112458>

13. Aftatah M, Zebbara K, Asri SE. Modeling Low-cost Inertial Navigation Systems and their Errors. *Int J Comput Netw Commun*. 2024 Nov;16(6):127-44. <https://doi.org/10.5121/ijcnc.2024.16608>
14. Yampolsky Z, et al. Multiple and Gyro-Free Inertial Datasets. *Sci Data*. 2024 Oct;11:1080. <https://doi.org/10.1038/s41597-024-03917-6>
15. Lyu P, Yong C, Lai J, Yuan C, Zhu Q, Han A. A Low-Altitude UAV Dataset Based on Visual-Inertial Navigation. *IEEE Sens J*. 2024 Dec;24(24):41904-11. <https://doi.org/10.1109/JSEN.2024.3488289>
16. Thalagala RG, De Silva O, Jayasiri A, Gubbels A, Mann GK, Gosine RG. MUN-FRL: A Visual-Inertial-LiDAR Dataset for Aerial Autonomous Navigation and Mapping. *Int J Robot Res*. 2024 Oct;43(12):1853-66. <https://doi.org/10.1177/02783649241238358>
17. Mahdi AE, Azouz A, Abdalla AE, Abosekeen A. A Machine Learning Approach for an Improved Inertial Navigation System Solution. *Sensors*. 2022 Jan;22(4):1687. <https://doi.org/10.3390/s22041687>
18. Černilová B, Kuře J. IMU Sensor for In-Situ 3D Movement Monitoring of Particulate Matter. *Eng Proc*. 2024;82(1):20509. <https://doi.org/10.3390/ecsa-11-20509>
19. Vighala J, Ramesh NVK, Devanaboyina VR, Reddy BNK. Attitude, Position and Velocity determination using Low-cost Inertial Measurement Unit for Global Navigation Satellite System Outages. In: 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT). 2021 Jun. p. 61-5. <https://doi.org/10.1109/CSNT51715.2021.9509605>
20. Wang L, Zhang Z, Sun P. Quaternion-Based Kalman Filter for AHRS Using an Adaptive-Step Gradient Descent Algorithm. *Int J Adv Robot Syst*. 2015 Sep;12(9):131. <https://doi.org/10.5772/61313>
21. Das S, Boberg B, Fallon M, Chatterjee S. IMU-based Online Multi-lidar Calibration. In: 2024 IEEE Intelligent Vehicles Symposium (IV). Jeju Island, Korea, Republic of: IEEE; 2024 Jun. p. 3227-34. <https://doi.org/10.1109/IV55156.2024.10588695>
22. Li Y, Wang L, Wang Z, Li X, Li J, Su SW. On-site scale factor linearity calibration of MEMS triaxial gyroscopes. *ArXiv*. 2024 <https://doi.org/10.48550/arXiv.2405.03393>
23. Seo YB, et al. Analysis of Gyro Bias Depending on the Position of Inertial Measurement Unit in Rotational Inertial Navigation Systems. *Sensors*. 2022 Jan;22(21):8355. <https://doi.org/10.3390/s22218355>
24. Aftatah M, Zebbara K. Detecting GPS Spoofing Attacks Using Corrected Low-Cost INS Data with an LSTM Network. *Int J Adv Comput Sci Appl*. 2024;15(11).
25. Wu F, Luo H, Jia H, Zhao F, Xiao Y, Gao X. Predicting the Noise Covariance With a Multitask Learning Model for Kalman Filter-Based GNSS/INS Integrated Navigation. *IEEE Trans Instrum Meas*. 2021;70:8500613. <https://doi.org/10.1109/TIM.2020.3024357>
26. Thokala VS. Utilizing Docker Containers for Reproducible Builds and Scalable Web Application Deployments.
27. Duan Y, Li H, Wu S, Zhang K. INS Error Estimation Based on an ANFIS and Its Application in Complex and Covert Surroundings. *ISPRS Int J Geo-Inf*. 2021 Jun;10(6):388. <https://doi.org/10.3390/ijgi10060388>

## FINANCING

The authors did not receive financing for the development of this research.

## CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

## AUTHORSHIP CONTRIBUTION

*Conceptualization:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Data curation:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Formal analysis:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Research:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Methodology:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Project management:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Resources:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Software:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Supervision:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Validation:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Display:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Drafting - original draft:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.

*Writing - proofreading and editing:* Mohammed Aftatah, Abdelhak Khalil, Khalid Zebbara.