KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

GROUP T LEUVEN CAMPUS

Implementing a Mobile Wi-Fi and IMU Positioning System

R&D: SPAI

Yousef MAS'AD

Giulio VITOLO

Mateo SAKR

Dunia TORNILA JICHI

Yusuf HUSSEIN

Supervisor: Valerio Lorenzoni

Technical paper submitted as part of the Master of Science in Engineering Technology: Electronics and ICT Engineering

Academic Year 2024-2025

KU LEUVEN

Faculty of Engineering Technology Group T Leuven Campus — Technical paper submitted as part of the Master of Science in Engineering Technology: Electronics and ICT Engineering
Academic Year 2024-2025

Implementing a Mobile Wi-Fi and IMU Positioning System

R&D: SPAI

Hussein Yusuf, Tornila Jichi Dunia, Sakr Mateo, Vitolo Giulio, Mas'ad Yousef

Master in Engineering Technology: Electronics and ICT Engineering, Faculty of Engineering Technology, Group T Leuven Campus, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium

Supervisor: Valerio Lorenzoni
Engineering Technology: Electronics and ICT Engineering, Faculty of Engineering Technology, Group T Leuven Campus, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium, valerio.lorenzoni@esat.kuleuven.be

ABSTRACT

Head-Tracking systems play a crucial role in various fields such as augmented reality, gaming, and immersive audio. It is therefore essential for these systems to be as accurate as possible, in order to provide immersive experiences for the users. Specifically in the context of spatial audio, knowledge of the positioning of the head allows the alignment of auditory experiences with visual stimuli. One example of that would be in VR environments, where sound sources must correspond to visual cues to create a realistic 3D audio experience. The current problem with most of the state of the art solutions, is that they are often complex, expensive, and computationally intensive. This ends up limiting their accessability. This project aims to tackle these problems by combining several signal processing algorithms to enhance a cost-effective, head-tracking system. We explore multiple approaches to estimating position and orientation, and we present a collected and labelled dataset of 110 minutes of motion. Our method acheieves average error of 2m and 35 degrees on our collected dataset.

1 INTRODUCTION

In the context of the Helixon project (Hybrid, efficient, and liquid interpolation of sound in extended reality), a vital part of a sound interpolation system is the head-tracking system through which the sound interpolation system derives its behavior from. As such, in this paper we intend to explore a specific research question:

RQ: How can head position and orientation be efficiently estimated and tracked?

In our investigation of this question, we will present our research on various methods for position and orientation estimation and our results and evaluations of the different methods. Furthermore, we will present the Helixon dataset collected during the course of the research, and explore the methods of data collection, preprocessing, calibration, and synchronization used in creating the dataset.

2 DETAILED PROBLEM REQUIREMENTS

2.1 Target Environment

The goal of the head-tracking system was to determine the position and orientation in the specific target environment of KU Leuven's Group T Campus's spiral walkway. During the course of our research, we received the ground plans for the building.

Figure 2.1: The target environment for this project, Campus GroupT's spiral walkway (Leuven, Belgium).

2.2 Sensors and MCUs

Our project used the Bosch BNO055, a 9-axis intelligent absolute orientation sensor, and the BMP390, a high-precision barometric pressure sensor, for motion and altitude tracking, respectively. The BNO055 integrates a gyroscope, accelerometer, and magnetometer with an onboard microcontroller that fuses sensor data, offering orientation accuracy of ±1 ° in stable conditions. [1] Its self-calibrating features reduced the complexity of manual calibration, though environmental interference occasionally introduced noise. The BMP390 ensured altitude measurements in the precision of ±0.03 hPa, thereby allowing for height estimations within a range of ±0.25 meters. [2]

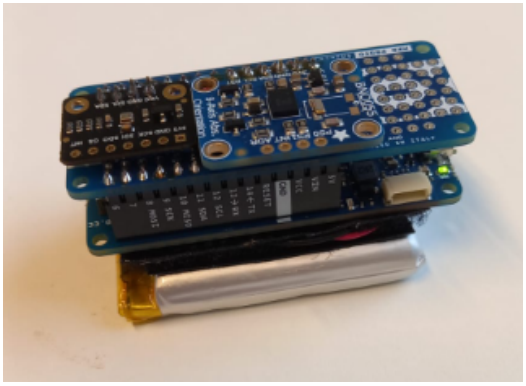

Figure 2.2: The Arduino module and sensors used.

3 RELATED WORK

The task of IMU-based head-tracking and positioning is a decades old task with significant literature exploring many different methods and solutions. A large portion of existing literature exploits knowledge of the system's behavioral model and multiple measurement sensors to implement Kalman filtering schemes [3, 4]. More recent methods also include step detection and counting methods in order to reduce drift issues in dead reckoning applications [5, 6]. Finally, with the modern advancements in compute and machine learning algorithms, machine learming methods have been implemented to create end-to-end solutions for position and orientation tracking using machine learning models and large labelled datasets. Additionally, using IMU data isn't the only proposed solution for cost-effective position tracking systems, with many system exploiting the ubiquity of WLAN/Wi-Fi to incorporate signal strength information into their system for additional positioning accuracy. [7, 8].

4 METHODS AND IMPLEMENTATIONS

4.1 Dataset Collection

In order to evaluate and test the various schemes discussed in this paper, we conducted controlled experiments to capture and label head position and orientation data along the spiral path. Two primary data sources were utilized: sensor measurements, including data from a 9 Degrees of Freedom (DOF) sensor and a barometric pressure sensor, and network signals captured from routers at the target environment.

4.1.1 Ground Truth Data

To measure accurate ground truth data, the RTAB-MAP application (Real-Time Appearance-Based Mapping) was used. RTAB-MAP is a Simultaneous Localization and Mapping (SLAM) framework that uses RGB-D, stereo, and LiDAR technologies, as well as graph-based incremental appearance-based loop closure detection. In general, SLAM involves emitting laser pulses, analyzing their return times, and generating 3D point clouds that represent the environment. The data can then be used to determine a subject's position and orientation. For more details, one can refer to the RTAB-MAP website [9] and Wevolver's LiDAR SLAM article. [10]

The process of collecting ground truth data involved initializing RTAB-MAP on an iPhone, starting a new data recording session, and walking along the designated spiral path. To make sure that the recorded data was as accurate as possible, the phone's camera was kept unobstructed throughout the recording. RTAB-MAP's output, which is sufficiently accurate for this project, was used as the benchmark ground truth data.

4.1.2 Sensor Data

During the collection of ground truth data, the Arduino setup was activated, and the subject walked along the spiral path to collect raw sensor data. By integrating and processing the collected raw measurements, the subject's orientation and position were estimated. The Wi-Fi data was collected using a 2nd Arduino microcontroller, which exclusively scanned for RSSI values. The data was retrieved by the main Arduino module via a UART connection.

4.1.3 Measurement Apparatus

To simulate real-world conditions, measurements were conducted with the sensors attached to headphones, replicating their intended final placement. A custom measurement device (fig. 4.1) was used to ensure accurate data collection. This device was then attached to the headphones as can be seen on fig. 4.1. With this approach, the collected data accurately reflected the end-use scenario, making it reliable for realistic position and orientation estimation.

4.1.4 Data Collection Process

After setting up the measurement device, the data was collected. The data collection was done on the spiral of the GroupT campus in Leuven, Belgium. For each data recording session, the person wearing the measurement device must start and end the session at the same point, more precisely at the beginning of the spiral. In addition, the person stops at the same point, more precisely at the top of the spiral, before heading back to the starting position. Every measurement session lasts 10 minutes, but extending the duration can improve accuracy.

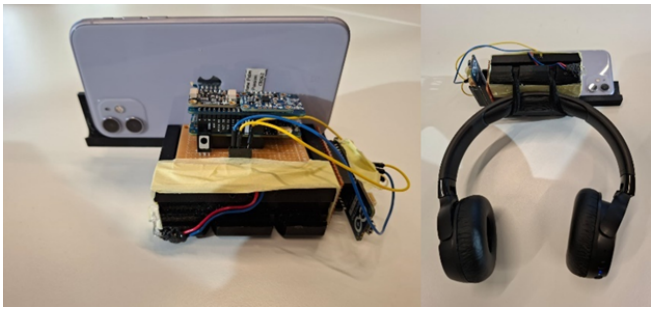

Figure 4.1: The 3D printed measurement apparatus.

Two types of measurements were taken. The first one represents an ideal sequence where the subject walks along the middle of the spiral in a straight line, with no interference, steady pace, and limited head rotation. The second type of measurement involves the person who is collecting data to walk along the spiral while applying more natural and random movements, such as moving up and down multiple times, moving the head in random orientations, moving away from the center of the spiral's path, etc. In total, 1 hour and 50 minutes of data was collected.

4.2 Dataset Preprocessing

After collecting our dataset, it had to be preprocessed in multiple ways before it was usable with our systems and evaluations. First of all, the data is sent from the Arduino to the desktop python environment where it's processed further.

4.2.1 Synchronization

Since the Arduino data and the ground truth data are running on two separate independently clocked systems, a synchronization step is required first. To do this, a computer vision-based synchronization mechanism was implemented.

As previously mentioned, both the raw data collection setup and the iPhone were mounted on a single measurement apparatus. An LED, connected to the Arduino, was positioned in the iPhone camera's field of view. Such LED lights up when the sensors connected to the Arduino are collecting data, serving as a synchronization marker. Us-

ing the OpenCV library, the irrelevant SLAM data that was recorded when the LED was off was filtered out. In addition, the sampling frequency of the raw data was much higher than the sampling frequency of the ground truth data, meaning that at the end of each recording session there were less ground truth samples than raw ones. Because of this, linear interpolation was used to match the oversampled raw data to the closest SLAM data points.

4.2.2 Alignment and Normalization

Due to the inconsistent starting orientations and initial conditions, the ground truth data isn't always aligned to the same origin. In order to ensure our evaluation was valid, we implemented a data preprocessing step that ensures the entire dataset is aligned in 3d space. The implementation first ensures all the spirals are centered about the same point, and then aligns the starting points of the aligned spirals.

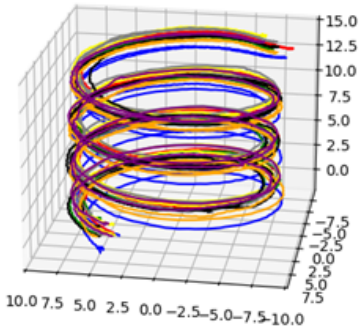

Figure 4.2: The aligned spiral data.

After the alignment step, the sensor data is normalized so that the initial measurements are the "baseline values". Orientation data has the initial measurement subtracted from all measurements, resulting in it starting at the orientation origin. The ground truth orientation is similarly processed, however in quaternion space. Finally, the pressure data is preprocessed to remove outlier values due to sensor errors.

4.3 Dataset Storage

After synchronization, the data is stored so that it can be further processed. More precisely, the data was stored in HDF5 format (Hierarchical Data Format version 5). More precisely, the data for each full measurement session is stored in one HDF5 file which consists of the ground truth data, the raw data, and the Wi-Fi data. HDF5 format was used for data storage due to its multiple advantages such as its ability to store large datasets in structured hierarchy. The format is presented below. [11]

Group / Category	Subcategories
GT_DATA	ORIENTATION
	POSITION
	TIMESTAMP
RAWDATA	9DOF
	BMP
	BNO
	PRESSURE
	RPY
	TIMESTAMP
WIFIDATA	BSSIDS
	COUNTS
	RSSIS
	TIMESTAMP

Table 4.1: Structure of an HDF5 file

4.4 Synthetic Data Generation

An additional synthetic data pipeline was implemented in order to achieve 3 main goals: providing an alternative to tedious data collection procedures, allowing for quick prototyping of algorithms and schemes before the full dataset was collected, and ensure behavioral integrity and reliability of implemented schemes before real evaluations. Since the required synthetic data generation was not meant to replace the actual dataset, we opted for a simple mathematical model for the sensor outputs, followed by sampled white gaussian noise.

The implemented pipeline is able to generate artificial sensor readings for any given "ground truth" trajectory, whether that trajectory is actual collected data, or synthetically generated positions. Using a simple discrete derivative formula, the velocities and accelerations were calculated from the input positions, and the rotations were calculated as the tangents to the synthetic spiral. Following that, the pressure value is calculated using the altitude-pressure formula [12] and white gaussian noise is added to the measurements.

Furthermore, artificial Wi-Fi RSSI data was added to the synthetic data pipeline, using defined BSSIDs in certain positions, and the distance-RSSI formula. [8] Finally, all this data is stored similarly to the actual dataset, in an identically formatted HDF5 sequence, to allow for interchangeability between the real and synthetic sequences.

4.5 Position Estimation

4.5.1 Naive Double Integration Dead Reckoning

The simplest and most well-known approach to positioning using IMU data is naive double integration dead reckoning (NDI) [13], where the acceleration components are discretely integrated into velocities, which are then in turn integrated into positions. The issue with this approach is that due to the double integration, sensor offsets, and noise can easily snowball due to the double integration procedure and as such the predictions drift very quickly. [13]. In experiments with NDI, the algorithm's predictions went out of control very quickly. Due to these issues, NDI is not a solution for the indoor positioning problem.

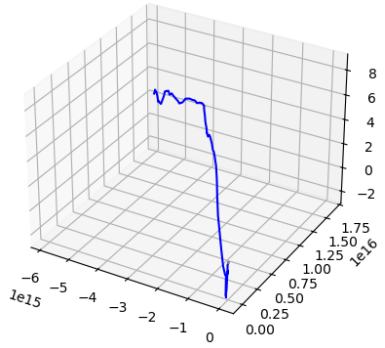

Figure 4.3: The massive drift issues apparent in NDI.

4.5.2 Pressure and Environmnt Model Based Positioning

The next positioning method we explored was using the BMP390 pressure sensor's pressure values, along with our environment model for the spiral environment.

Pressure Based Altitude Estimation

Using the following formula, we determined the height values using the pressure data received.

$$P(z) = P_0 e^{-\alpha z}$$

Using the derived pressure values, we calculate a 3D position in the target environment using a direct mapping from altitude values to 3D positions built on information about the target environment. Using information from both our ground truth data in our collected dataset, and ground plan data of the target environment. The mathematical model implements a mathematically ideal spiral, with its parameters tuned to best fit the ground truth data and the floor plan data. After visual tuning and comparing values with the ground plans, the parameters finally used are shown below, along with a diagram showing the ground truth data next to the model data.

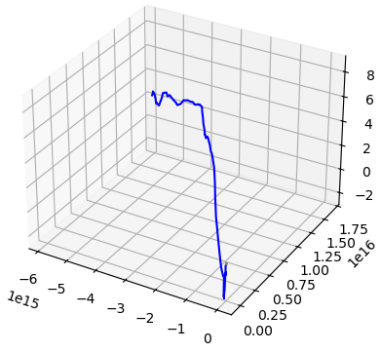

Figure 4.4: A visualization of the derived spiral model (blue) and the ground truth data (gray).

Parameter	Value (m)
`spiral_pitch`	4.2
`spiral_radius`	8.0
`path_width`	2.4

Table 4.2: Parameters of the spiral model.

The spiral mathematical model was furthermore developed with functionality to align them with ground truth data spirals, to allow for more accurate inference. This is done via calculating the maximum bounds along the x and y axes of the spiral, and then using the resulting centroid as the center of the spiral. Furthermore, the spiral model is capable of deriving a 3D position using a Z value using the formulas below.

$$\theta = 2\pi\frac{z}{p}$$

$$x = x_0 + r \cdot cos(\theta + \varphi)$$

$$y = y_0 - r \cdot sin(\theta + \varphi)$$

Where theta is the angle on the circular top-down projection of the spiral, x, y, z are the 3D positions of the output point (and z is the input height), phi is the phase shift of the spiral calculated at alignment, r is the radius of the spiral, p is the pitch of the spiral, and x_0 and y_0 are the center x and y position of the spiral's top-down projection. Additionally, the model is capable of calculating the closest point on the spiral model to an arbitrary input point. The distance between an arbitrary point and a point on the spiral was derived mathematically in cylindrical space, using r_{sp}, h_{sp}, and θ_{sp} as the position of the closest point on the spiral to the arbitrary point r_{pt}, h_{pt}, and θ_{pt}. Then an optimization problem was solved by solving $d^2\prime = 0$. The formulas for this problem are shown below

$$d^2 = r_{sp}^2 + r_{pt}^2 - 2r_{sp}r_{pt}cos(\theta_{sp} - \theta_{pt}) + (h_{sp} - h_{pt})^2$$

$$\theta_{sp} = 2\pi\frac{h_{sp}}{p}$$

$$\frac{d(d^2)}{d(h_{sp})} = \frac{4\pi r_{sp} r_{pt}}{p} sin(\frac{2\pi h_{sp}}{p} - \theta_{sp}) + 2h_{sp} - 2h_{pt}$$

After the formula above was derived, finding a solution analytically was difficult, and solvers were unable to compute a solution due to the periodicity of the sine section of the formula. As a result, we opted for a polynomial half period approximation of a sine wave and used a symbolic solver to derive the candidate solutions for the optimal point. There are 3 candidate solutions, we choose the solution that produces the lowest distance and return the point at the resulting h_{sp}.

$$sin\ x \approx \frac{16x(\pi - x)}{5\pi^2 - 4x(\pi - x)}$$

4.5.3 Wi-Fi Based Positioning

Given the ubiquitous nature of Wi-Fi networks, using Wi-Fi data for positioning is a very attractive approach. Additionally, the target environment, being a university campus, was filled with constant position Wi-Fi networks, which could be used as anchors for a fingerprinting system. To this end, we implemented a random forest regressor model and trained it over our dataset in order to predict 3D positions using the detected BSSID to RSSI data measured from the Arduino's Wi-Fi module. [14]

The model's input format is a vector of RSSI values, with each position being associated with a unique Wi-Fi BSSID found in the training set. During inference, any BSSIDs not found in the dataset are disregarded, and any BSSIDs found have their associated RSSI values placed at the corresponding position in the input vector. Any undetected BSSIDs in the input vector are set to a value of -100dB.

A grid search was conducted over the number of estimators and the maximum depth, and the following model hyperparameters were determined to perform best for our task.

Parameter	Value
bootstrap	True
ccp_alpha	0.0
criterion	'squared_error'
max_depth	28
max_features	1.0
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
monotonic_cst	None
n_estimators	525
n_jobs	None
oob_score	False
random_state	None
verbose	0
warm_start	False

Table 4.3: Model Hyperparameters.

4.5.4 Kalman Filtering Schemes

Using a combination of the IMU's accelerometer data with the NDI method, and the BMP390's pressure data and the derived altitudes, we implemented our first Kalman filtering scheme, which fuses the data from these two sources to attempt to find a more accurate output. The Kalman filter was implemented with a state vector containing the positions and velocities along the x, y, and z axes. A mathematical formulation of the filter is shown below. These filters were then tuned by a scalar parameter scaling these matrices.

$$H_{\text{PRESSURE}} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Matrix 1: Measurement matrix for pressure state variables.

$$H_{\text{WIFI}} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Matrix 2: Measurement matrix for WiFi state variables.

$$A = \begin{bmatrix} 1.0 & 0.0 & 0.0 & \Delta t & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & \Delta t & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & \Delta t \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Matrix 3: State transition matrix.

$$B = \begin{bmatrix} 0.5\Delta t^2 & 0.0 & 0.0 \\ 0.0 & 0.5\Delta t^2 & 0.0 \\ 0.0 & 0.0 & 0.5\Delta t^2 \\ \Delta t & 0.0 & 0.0 \\ 0.0 & \Delta t & 0.0 \\ 0.0 & 0.0 & \Delta t \end{bmatrix}$$

Matrix 4: Control transition matrix.

The filter's covariance and noise matrices were tuned by hand and evaluated over dataset sequences in order to choose the best performing values.

Another architecture we experimented with was incorporating the Wi-Fi positioning data as well into the Kalman filtering scheme. This Kalman filter used the same state vector and transition matrix, however, the filter was updated using both the pressure-altitude based positions and the Wi-Fi positioning data.

4.6 Orientation Estimation

The second step of the head-tracking system was implementing an attitude estimation system. Our implemented attitude system uses the BNO055 IMU's orientation estimation system, with some further postprocessing. The data is retrieved as roll, pitch, yaw Euler angles in degrees, and are baselined on an initial assumption of the user starting approximately at the orientation origin. [**?**]

4.7 Real-Time Communication

The final step of the head-tracking system was ensuring the system can run in real-time and not just on the collected dataset. Given that our signal processing schemes were implemented to run in a python desktop environment and were incapable of running on the microcontroller, we had to set up real-time network communication between the system and the python environment. We implemented two interchangeable communication layers; one built on TCP, and one built on UDP. The UDP system is preferred and was used for our data collection and real-time testing, since the system is more efficient and robust to packet drops. The system relies on two internal data structures used to represent the sensor data and the Wi-Fi data respectively.

Type	Name	Size (bytes)	Notes
`unsigned long`	`microsT`	4	4 bytes padding
`double`	`linaccelx, linaccely, linaccelz`	$8 \times 3 = 24$	Linear acceleration
`double`	`gyrox, gyroy, gyroz`	$8 \times 3 = 24$	Gyroscope values
`double`	`magnx, magny, magnz`	$8 \times 3 = 24$	Magnetometer values
`double`	`roll, pitch, yaw`	$8 \times 3 = 24$	Orientation values
`int8_t`	`tempnbo`	1	7 bytes padding
`double`	`tempbmp`	8	Temperature
`double`	`pressure`	8	Pressure

Table 4.4: Structure of `DataEntry` (Size: 128 bytes).

Type	Name	Size (bytes)	Notes
`unsigned long`	`microsT`	4	-
`int8_t`	`rssiCnt`	1	1 byte padding
`byte`	`BSSIDs[25][6]`	$25 \times 6 = 150$	1 byte padding
`int32_t`	`RSSIs[25]`	$25 \times 4 = 100$	-

Table 4.5: Structure of `RssiDataEntry` (Size: 256 bytes).

4.8 Visualization

To visualize data in real-time, we initially used matplotlib's Animation module for simple animations to validate our solution. However, due to its limitations, we switched to Blender, a free 3D graphics software known for realistic 3D rendering, camera, lighting, and material options, and Python scripting capabilities. We first animated the ground truth and unfiltered data in Blender, using the data from h5 files. Once confirmed to be working as expected, we modeled the real-time data from our implementation.

Blender includes its own Python environment, requiring the installation of necessary Python libraries within Blender's Python. Since the filtering script and Blender visualization ran in separate Python environments, we established communication between them using TCP sockets. The main Python script acted as a server, sending orientation and position data to the Blender script, which acted as a client, receiving the data.

We modeled the spiral, floor, and walls in 3D, adding lighting and camera tracking. In the final Blender file, a face is visible in the main frame, adjusting the camara to follow along the spiral as a person ascends. Blender on our PCs could render at a maximum of 25 frames per second (fps), while our solution sampled at 100Hz. To synchronize the visualization with real-time position and orientation, we sent only one sample of position and orientation data out of every four samples to the Blender script. fig. 4.5 displays the final Blender animation. On the right is the camera view, focusing on the monkey face at the center. On the left is a overview visualization of all the objects that were modeled.

5 RESULTS AND EVALUATION

In order to assess the quality and performance of the proposed systems, we ran various qualitative and quantitative evaluations. The results of these evaluations are presented below.

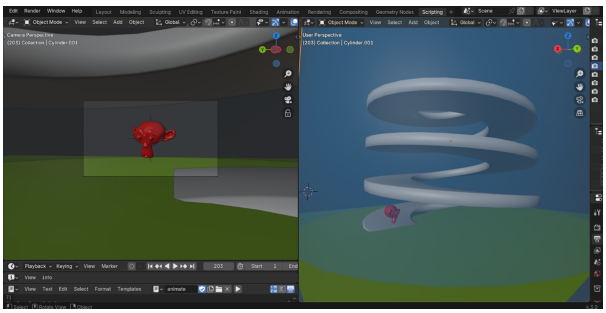

Figure 4.5: A preview of the implemented visualizer.

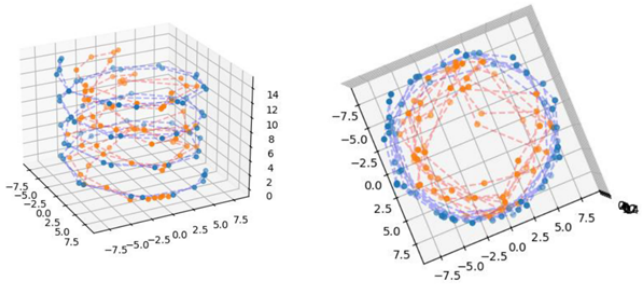

Figure 5.1: An example of the Wi-Fi RFR Model's predictions (orange) and the ground truth path (blue).

5.1 Hardware Limitations

Our first plan was to run the Kalman filter on the Arduino itself. This proved challenging since the filter requires a lot of computation and we had doubts on the performance of Arduino with the filter on producing real time results. To that end, we implemented a simple Kalman filter that did not integrate all the aspects of the real filter in order to address the hardware limitation, which worked well but did not achieve our expected performance or results. The Arduino MKR 1010 is not designed for computationally intensive tasks like matrix multiplication. Small matrices (e.g., 3x3 or 4x4) can be handled reasonably, but as matrix sizes grow, performance drops significantly due to limited processing power and memory. NumPy uses highly optimized libraries like BLAS and LAPACK. These libraries leverage multi-threading, SIMD instructions, and efficient memory access patterns to maximize speed. The performance scales well with larger matrices.[4] To that end, we decided to implement the complete Kalman filter in Python and we would transfer the sensor reading from the Arduino to a Laptop to process it.

5.2 Positioning Systems

5.2.1 Wi-Fi Data Random Forest Regressor

The Wi-Fi RFR model was evaluated on average error, max error, mean squared error, and R2 score in order to show various aspects of the model's behavior.

Table 5.1: Error Metrics for Test and Unseen Sets

Metric	Test Set	Unseen Set
Average Error (m)	6.47	6.93
Max Error (m)	9.27	11.33
Mean Squared Error (m^2)	43.9	49.7
R^2 Score	0.536	-

A qualitative sample of the RFR model running on a single data sequence is seen below, showing the Wi-Fi model's predictions in orange, and the ground truth positions in blue.

5.2.2 Kalman Filtering Schemes

In the evaluation of the positioning system, three main metrics were used: the absolute trajectory error (ATE), relative trajectory error (RTE), and the average error. [9, 13] The implemented evaluations were implemented with a linear interpolation system, in order to be able to evaluate the metrics over two signals with different sample times.

Table 5.2: Trajectory and Average Errors Using Different Methods

Metric	Pressure Only	Kalman Fusing Accelerometer and Pressure	Kalman with Wi-Fi Data
ATE (m)	3.13	3.13	5.49
RTE (m)	2.81	2.80	2.87
Average Error (m)	2.14	2.14	4.06

5.3 Orientation Estimation System

In our evaluation of the orientation system, a preprocessing step was conducted in order to improve the synchronization of the ground truth and measured data. A cross correlation was calculated between the two signals and used to determine an optimal time shift between the two signals. Even with the rough synchronization applied during data collection, the cross correlation was still able to detect a time shift, which was accounted for in this evaluation.

Table 5.3: Error Metrics in Degrees

Metric	Value (°)
Median Error	28
Average Error	35
Max Error	168

Furthermore, a qualitative sample of a single sequence is shown below. In this sequence, the heading (roll) is represented by the red signals, and the pitch and yaw are represented as the

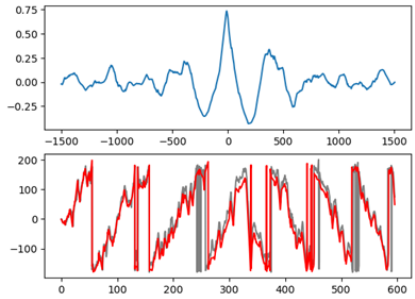

Figure 5.2: The results of the cross correlation of the ground truth and measured rotations (top) and the shifted data (red) after the delay was calculated (bottom).

Table 5.4: Error Metrics for Roll, Pitch, and Yaw

Metric	Roll (°)	Pitch (°)	Yaw (°)
Median Error	27.4	3.59	2.42
Average Error	33.96	3.85	2.78
Max Error	168.36	13.97	11.67

blue and gray signals.

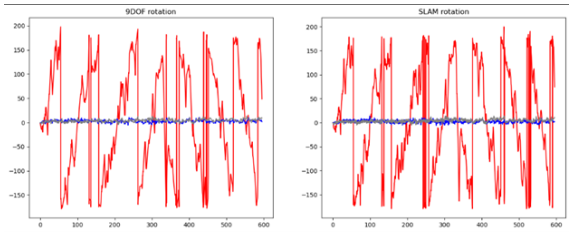

Figure 5.3: The measured (left) and ground truth (right) rotation values with heading, pitch, and yaw in red, gray, blue respectively.

6 DISCUSSION

6.1 Positioning Systems

In terms of pure performance, the pressure-based system heavily outperforms every other proposed system, barring the Kalman filtering schemes tuned to prioritize pressure data. This is due to the high accuracy and sampling rate of the pressure data in our system, which when combined with the well-tuned altitude based mathematical spiral model, allows us to calculate accurate positions at a very high sampling rate. Due to this, the noisy data from the accelerometers and the Wi-Fi RFR weren't able to improve the predicted positions significantly. However, the constraints placed on the system in terms of the target environment are difficult to generalize to other use-cases, where a system relying only on pressure data wouldn't be able to determine the X and Y positions with a reasonable accuracy.

Furthermore, if the spiral model used was mistuned or if the alignment was imperfect, the accuracy of the pressure-based system suffers significantly, which we encountered during our tuning. In cases where the environment model cannot deterministically predict an X, Y position for a given height, or in cases where the environment model is inaccurate or inconsistent, the Wi-Fi based Kalman filtering scheme can provide an additional layer of error correction and outperforms the pressure-only system.

6.2 Orientation Estimation Systems

The orientation system proposed appears to perform well from a distance, with generally accurate orientation values along entire sequences. However, the system is prone to occasional drift issues, specifically on the heading axis. This is due to two main reasons. Firstly, the heading axis lacks an accurate baseline in the form of the gravity vector, which is only available on the other two axes. This issue can be mitigated via integrating a more accurate sensor or implementing a Kalman filtering/machine learning scheme to further improve the predicted angles. Secondly, the heading axis exhibits significantly larger variance than the other two axes of rotation, simply due to human anatomy and range of motion. However, the orientation estimation system generally performs well, with below 30 degrees of median angular error.

7 FUTURE WORK

7.1 Sensors and MCUs

A proposed alternative to the Arduino MKR1010 is an upgrade to an ESP32 device. An ESP32-WROOM-32 upgrade offers serious improvements over the Arduino MKR1010 in applications needing efficient processing, reduced latency, and energy savings. ESP32 outperforms MKR1010 with a dual-core 240 MHz processor, as opposed to MKR1010's single-core 48 MHz, and a larger RAM capacity of 520 KB against 32 KB. The increased computational power of the ESP32 enables it to do some real-time computations, like running Kalman filters, onboard without the use of an external computer. This removes the high latency and network dependency associated with the MKR1010-based method, where sensor data has to travel to a laptop for processing.

The ESP32 also boasts significantly improved connectivity capabilities that include Wi-Fi and Bluetooth v4.2/BLE. With its dual-mode Wi-Fi operation (STA + AP), the ESP32 allows asynchronous network scanning without interrupting the active connections, which the MKR1010 cannot do. This allows for continuous data transmission while keeping the power consumption low by using intermittent Wi-Fi.

Also, it reduces the system complexity since the ESP32 eliminates the use of external PCs, making deployment and integration easier. The performance and scalability increase because of its capability to handle real-time updates on its own. At a lower cost of $5-10 with more I/O pins, 36 compared to 22, the ESP32 presents an economical, high-performance solution.

Among several conclusions, it is considered that the migration to the ESP32-WROOM-32 avoids limitations concerning latency, energy usage, and system complexity brought into perspective by MKR1010, adding computational efficiency that makes this board definitely superior for real-time application employment.

[15]

7.2 Synthetic Data Generation

In order to improve the quality of our synthetic data so it better mimics real life data, there are several things that can be improved. Firstly, the velocity equation used, and trajectory could be improved to be more complex and to be better mimic real-life velocity and trajectory of a person going up a spiral. However, since these are modeled by mathematical equations, they cannot model the random behavior of human behavior.

Another adjustment would be to better model the noise added by the different sensors. In the datasheet, the average and the standard deviation of said noise is mentioned because it can be modeled as additive white gaussian noise, however in real life noise generated by the sensor might be more complex. To add noise to our synthetic data that can more accurately copy real-life noise created by the sensors, deep learning can be used.

Generative AI can help create realistic sensor noise. By training on a dataset collected from still sensors - where acceleration is zero and pressure, gravity, and the magnetic field are constant - Generative AI could replicate the noise profile. To account for variability due to factors like temperature or sensor orientation, data collection should be repeated on different days and in different positions. [16] describes creating a synthetic IMU dataset simulating a person falling. The dataset was trained using SLAM data recorded in an environment free from occlusions. Similarly, we could collect SLAM and IMU data to isolate sensor noise and use it to train a model capable of generating highly realistic IMU noise. In fig. 7.1 the structure of the LSTM model used to generate synthetic data can be seen and fig. 7.2 shows the experimental set up used.

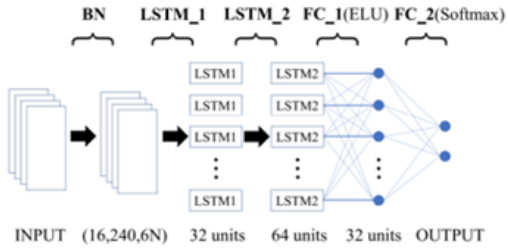

Figure 7.1: The LSTM architecture used for generating the synthetic data.

7.3 Data Collection and Processing

For the future there are several points to improve upon both for data collection and pre-processing. For Data collection:

- Data collection could be done using more accurate SLAM tools instead of an iPhone. An example of such a tool would be a high-quality stereo camera with LIDAR. This would help improve the accuracy of the data measured.
- Generate synthetic variations of the collected dataset to simulate different scenarios. So, using data augmentation, we would create different datasets from an originally measured one to increase and vary our dataset.

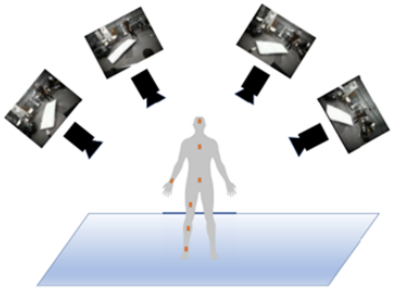

Figure 7.2: The experimental setup used for generating the synthetic data.

For pre-processing:

- As mentioned before, the method used for synchronization of the ground truth and the raw data was not perfect as it was coarse tuning. Therefore, a fine-tuning method can be applied. This first starts by resampling one of the two signals so that they both have the same sampling frequency. This is crucial because the next step is performing the cross correlation of the signals, and to do that, the signals should have a constant and equal sampling frequency. Performing the cross correlation informs us of how much the 2 signals lag each other, and we can then shift one of the signals by that time to match them. This will help synchronize the signals properly. [13]

7.4 Position Estimation

The implemented positioning system can be improved via various methods, however, given the scope and requirements of the project, we propose a few main future improvements to the system.

Firstly, a major issue facing the positioning system, is the inaccuracies of the mathematical spiral model. In order to improve this aspect of the system, a machine learning model can be trained in order to predict positions along the spiral more accurately than a constant mathematical model, and can better reflect the inconsistencies and intricacies of the ground truth data.

Secondly, another issue the system faces, is the lack of usefulness of the accelerometer data due to the high noise levels which causes excessive drift. To combat this, a common method explored in the literature is step counting, which employs situational assumptions about human movement in order to more accurately apply dead reckoning based positioning using IMU data. To implement this, an imu would need to be rigidly attached to the foot of the user ideally, however if this was infeasible, methods attempting step counting using head-attached IMUs can be explored. [6, 5] Thirdly, the implemented simple Kalman filtering scheme fails to capture non-linear behavior exhibited by the system, and so implementing an Unscented Kalman filter [17] can improve the performance of the system by

improving the system's robustness to non-linear effects.

Finally, a trivial improvement which can be implemented to increase the performance of the overall system, is to replace the used sensors with better performing consumer grade sensors. As an example, the BNO085 is a similar IMU to the used BNO055 with better performance, and within the same price range.

7.5 Orientation Estimation

Future work in terms of the orientation estimation mainly should focus on improving the heading axis accuracy. The heading axis faces the largest issues in performance as seen in our evaluations, which is due to the two reasons discussed earlier. Resolving the issues due to the lack of a baseline vector in the form of the gravity vector will mainly involve implementing a more accurate baseline system based on magnetometer data or integrating different sensors that allow for accurate measurements on the heading axis. An example of this could be integration of spatially spaced sensors, and employing time-difference-of-arrival (TDoA) methods to measure the heading angle. [7] Furthermore, machine learning methods can be employed in order to improve prediction fidelity, and to learn non-trivial patterns from the dataset. [13]

8 CONCLUSION

In summary, the project successfully developed a robust real-time position and attitude estimation system tailored for the constrained indoor environment of KU Leuven's Group T Campus spiral walkway. The work implemented several techniques, such as altitude-based pressure models, Kalman filtering schemes, and machine learning approaches like Random Forest Regression, by fusing sensors with the Bosch BNO055 IMU, BMP390 pressure sensor, and Wi-Fi-based positioning. While the pressure-based system was the most accurate for vertical positioning, the Wi-Fi integration and Kalman filters did improve the accuracy in complex conditions. The attitude estimation, while reliable, had issues such as drift in the heading axis because of the inherent limitations in the IMU-based measurements.

Future work is directed at both hardware and algorithm improvements. For example, moving to more powerful microcontrollers, such as the ESP32, would support on-device processing and decrease latency and system complexity. Other improvements involve refining the generation of synthetic data, enhancing synchronization during preprocessing, and using higher-order positioning techniques, such as step counting and machine learning-driven spiral alignment. These improvements are aimed at solving current limitations, enhancing generalizability, and ensuring the robustness of the system in various real-world applications.

ACKNOWLEDGEMENTS

- Prof. Toon Van Waterschoot, for presenting us with various useful concepts guiding us towards a starting point and for his insightful explanations and informative reference material.
- PhD. Valerio Lorenzoni, for the continuous support, feedback, and encouragement during the course of our work on the project.
- Prof. Koen Eneman, for providing us with the basis for many concepts and ideas we needed through the course of this project, and for his excellent reference material on the subject matter.

BIBLIOGRAPHY

[1] "Bno055 datasheet." [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf

[2] "Bmp390 -datasheet." [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp390-ds002.pdf

[3] A. M. Sabatini, "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation," *Sensors*, vol. 11, pp. 9182–9206, 09 2011.

[4] H. Hellmers, A. Norrdine, J. Blankenbach, and A. Eichhorn, "An imu/magnetometer-based indoor positioning system using kalman filtering," 10 2013.

[5] S. Tiwari and V. K. Jain, "A novel step detection technique for pedestrian dead reckoning based navigation," *ICT Express*, 10 2022.

[6] L. Huang, H. Li, W. Li, W. Wu, and X. Kang, "Improvement of pedestrian dead reckoning algorithm for indoor positioning by using step length estimation," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-3/W1-2022, pp. 19–24, 10 2022. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLVIII-3-W1-2022/19/2022/isprs-archives-XLVIII-3-W1-2022-19-2022.pdf

[7] I. Stanculeanu and T. Borangiu, "Enhanced rssi localization system for asset tracking services using non expensive imu," *IFAC Proceedings Volumes*, vol. 45, pp. 1838–1843, 05 2012.

[8] F. Shang, W. Su, Q. Wang, H. Gao, and Q. Fu, "A location estimation algorithm based on rssi vector similarity degree," *International Journal of Distributed Sensor Networks*, vol. 10, p. 371350, 01 2014.

[9] "Rtab-map," RTAB-Map. [Online]. Available: https://introlab.github.io/rtabmap/

[10] S. Malik, "Lidar slam: The ultimate guide to simultaneous localization and mapping," www.wevolver.com, 05 2023. [Online]. Available: https://www.wevolver.com/article/lidar-slam

[11] “Hierarchical data formats - what is hdf5? — nsf neon — open data to understand our ecosystems,” www.neonscience.org. [Online]. Available: https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5

[12] “Barometric pressure experiments.” [Online]. Available: http://boson.physics.sc.edu/Academics/UStudies/doc/CourseInfo/PHYS101L/Exp.12Barometric_Pressure.pdf

[13] H. Yan, S. Herath, and Y. Furukawa, “Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods,” arXiv.org, 2019. [Online]. Available: https://arxiv.org/abs/1905.12853

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Ã. Duchesnay, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825 – 2830, 2011. [Online]. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[15] J. Brownlee, “Numpy multithreaded matrix multiplication (up to 5x faster) - super fast python,” Super Fast Python, 05 2023. [Online]. Available: https://superfastpython.com/numpy-multithreading-matrix-multiplication/

[16] J. Tang, B. He, J. Xu, T. Tan, Z. Wang, Y. Zhou, and S. Jiang, “Synthetic imu datasets and protocols can simplify fall detection experiments and optimize sensor configuration,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 1–1, 01 2024.

[17] P. Bernal-Polo and H. MartÃnez-BarberÃ¡, “Kalman filtering for attitude estimation with quaternions and concepts from manifold theory,” *Sensors*, vol. 19, p. 149, 01 2019.

[18] “Bhaskara i’s approximation to sine,” Archive.org, 2019. [Online]. Available: https://web.archive.org/web/20120316083451/http://www.new.dli.ernet.in/rawdataupload/upload/insa/INSA_1/20005af0_121.pdf

[19] G. Welch and G. Bishop, “An introduction to the kalman filter,” 2006. [Online]. Available: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[20] E. , “Esp32 series datasheet,” 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[21] e. , “Esp32-wroom-32 datasheet.” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[22] R. Gomes, M. Ahsan, and A. Denton, “Random forest classifier in sdn framework for user-based indoor localization,” *Electro/Information Technology*, 05 2018.

[23] H. LeppÃ¤koski, J. Collin, and J. Takala, “Pedestrian navigation based on inertial sensors, indoor map, and wlan signals,” *Journal of Signal Processing Systems*, vol. 71, pp. 287–296, 11 2012.

[24] G. Bradski, “The opencv library,” *Doctor Dobbs Journal*, vol. 25, 11 2000. [Online]. Available: https://www.researchgate.net/publication/233950935_The_Opencv_Library

[25] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del RÃo, M. Wiebe, P. Peterson, P. GÃ©rard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with numpy,” *Nature*, vol. 585, pp. 357–362, 09 2020.

APPENDICES