UNIVERSITÄT ZU LÜBECK

MDS4AGT
Indoor Localisation
SS25

Scenario 2:
Indoor Localisation in Python

Sara Farshi, Narges Shafieyoun, Yasser Ibourk

23.06.2025

Content

1. Introduction

Indoor localisation is the process of determining the position and orientation of a person or device within a building, where traditional GPS signals are unreliable or entirely unavailable. This task poses significant challenges due to complex environments, sensor noise, and the lack of absolute position references. In recent years, smartphones have become increasingly central to this field due to their built-in inertial measurement units (IMUs), which include accelerometers and gyroscopes.

The goal of this project is to implement a complete pedestrian dead reckoning (PDR) pipeline using only the inertial sensors available on a smartphone, without relying on external infrastructure such as WiFi, Bluetooth, or floor plans. This constraint aligns with the second scenario of the MDS4AGT course project, emphasizing a lightweight and fully self-contained solution for trajectory estimation.

Inspired by the probabilistic sensor fusion principles outlined in the lectures by Prof. Dr. Frank Deinzer and Prof. Marcin Grzegorzek, the system models a pedestrian's state over time, comprising position and heading, by incrementally estimating movement from acceleration peaks (steps) and integrating gyroscopic angular velocity (turns). While complex recursive filtering methods like Kalman or particle filters are commonly used in research, this project focuses on the essential step–by–step estimation components: data acquisition, preprocessing, step detection, heading estimation, and path reconstruction.

The resulting pipeline enables the approximation of the user's trajectory within a 2D indoor space by converting inertial motion into spatial coordinates, using a series of transformations and integrations. Although limited by drift and sensor errors, this method offers a practical and extensible foundation for more sophisticated probabilistic indoor positioning systems.

2. Data acquisition

To begin the localisation process, motion data was collected using the Phyphox [1] smartphone application, which allows precise logging of sensor values such as acceleration and angular velocity in CSV format. The dataset includes six recording sessions, each consisting of two files: one for accelerometer readings and one for gyroscope readings. Each file captured time-stamped values along three axes (x, y, z), as well as the computed magnitude of the vector norm.

During data acquisition, we adhered to the predefined data acquisition protocol. In particular, we followed the instruction to record accelerometer data while keeping the phone in a (somewhat) fixed orientation for a few seconds before and after the actual walking motion. Consequently, after loading the raw data, we removed the first and last 10 seconds of each recording, as these segments corresponded to stationary periods not relevant to the localisation task.

As part of preparation, the relevant time intervals were extracted by applying time-based masks, and the corresponding accelerometer and gyroscope data were isolated for further analysis. This ensured temporal consistency between both sensor modalities, which is essential for accurate fusion and integration in later stages of the pipeline.

3. Step detection using accelerometer data

The process of step detection from the prepared accelerometer data can be divided into three key components: noise removal, peak detection, and step start estimation. Since the subsequent stage, namely *angular integration using gyroscope data*, is relatively straightforward and offers limited room for variation, we focused our efforts on experimenting with different techniques in this step. Specifically, we tested multiple options for signal smoothing, peak detection, and step start estimation. The methods used were guided by suggestions presented in the accompanying lecture material.

a. Noise removal

Because raw inertial data is often affected by sensor noise, user-induced motion artifacts, or hand tremors, effective denoising is crucial to ensure reliable step detection. In this context, the first step was to compute the total acceleration magnitude by applying the Euclidean norm to the x, y, and z components. Although Phyphox already provides a precomputed absolute acceleration value, recalculating the magnitude gave us full control over preprocessing and consistency across different data sources.

To suppress unwanted high-frequency components, we tested and compared three filtering approaches:

- A Butterworth low-pass filter, implemented with a cutoff frequency of 3 Hz and fourth-order design, to attenuate high-frequency noise.
- A simple moving average filter, which smooths local fluctuations using a sliding window.
- A Savitzky-Golay [2] filter, which performs polynomial smoothing while preserving signal shape and peak locations.

Each of these filters was applied to the computed acceleration magnitude, producing a denoised signal suitable for peak-based step identification.

b. Peak detection

Following the smoothing step, the goal was to identify the characteristic peaks in the filtered acceleration magnitude signal that correspond to individual footsteps. The underlying assumption is that each human step generates a prominent positive acceleration spike, which can be detected reliably if the signal is sufficiently clean.

We implemented two alternative methods for detecting peaks:

- The standard `scipy.signal.find_peaks()` function, configured with a minimum peak distance of 20 samples and a minimum prominence of 0.3, to ensure robustness against noise and avoid multiple detections per step.
- A custom implementation of smoothed z-score peak detection, which dynamically tracks mean and standard deviation over a sliding window and marks significant deviations as potential peaks. This method allowed for an adaptive detection threshold based on the local statistical behavior of the signal.

c. Step start estimation

Once peak timestamps were extracted, we explored different strategies to estimate the exact timing of step onset, which is critical for aligning movement phases with orientation data in the localisation pipeline.

We implemented three variants:

- A simple baseline approach that uses the timestamp of the first detected peak.
- A smoothed alternative that estimates step start as the average of the first two peak timestamps, mitigating the effect of noise or jitter.
- A physiologically motivated method that determines step start by locating the minimum acceleration point between the first two peaks, under the assumption that the lowest point precedes the push-off phase of the step.

These variants allowed us to evaluate the robustness of heading estimation in relation to the timing accuracy of step events.

4. Angular integration using gyroscope data

After detecting individual steps, the next phase in pedestrian dead reckoning (PDR) involves estimating the user's heading direction. This is done by integrating angular velocity data from the gyroscope over time. The process relies on combining accelerometer and gyroscope information to estimate the user's orientation in the horizontal plane and can be divided into two main steps: computing the rotation matrix and integrating the projected angular velocity.

a. Find rotation matrix R(t)

To interpret the gyroscope readings in the global coordinate frame, a rotation matrix R(t) is computed for each time step based on accelerometer data. Since the accelerometer also senses gravity, we can extract an estimate of the gravity vector by applying an exponential moving average filter to smooth out transient motion artifacts. The function `compute_rotation_matrices()` performs this smoothing using a factor `mu` and constructs a local coordinate frame at each timestamp.

Specifically, the smoothed gravity vector is used to define the z-axis of the world frame (u_z). The x-axis (u_x) is obtained by taking the cross product of u_z with a fixed reference vector (typically the global y-axis), and the y-axis (u_y) is computed as the cross product of u_z and u_x. These three orthogonal vectors form the columns of the rotation matrix R(t), which maps vectors from the device's local coordinate system into the global frame.

b. Estimating turning angle by integrating projected turn rate

With the rotation matrix R(t) at each time step, the angular velocity vector measured by the gyroscope can be projected into the world frame. Of particular interest is the z-component of this transformed vector, which corresponds to the rotation around the vertical axis and thus captures changes in the user's heading.

The function `integrate_angle()` performs this integration. First, the time differences Δt between gyroscope samples are computed. Then, the z-component of the angular velocity is averaged across time steps and cumulatively integrated to estimate the heading angle θ(t) in radians. This heading information is aligned with the timestamps of detected steps and is used in the final trajectory reconstruction.

5. Plotting estimated tracks

Trajectory estimation represents the final step of the PDR pipeline, where previously detected step timestamps and their associated heading angles are combined to reconstruct the user's two-dimensional walking path. This is implemented in the `compute_path()` function, which calculates the x- and y-position incrementally, starting from the origin at (0, 0). For each detected step, the new position is computed using the heading angle θ_t at time t and a fixed step length λ according to:

$$x_{t+1}=x_t + \lambda \cdot cos(\theta_t), \qquad y_{t+1}=y_t + \lambda \cdot sin(\theta_t)$$

To obtain θ_t, the angular velocity data from the gyroscope is first projected into world coordinates using the rotation matrices derived from filtered accelerometer data. The z-component of the transformed angular velocity is then integrated over time via `integrate_angle()` to yield the continuous heading angle trajectory.

The heading at each step is extracted by interpolating the integrated angle at the respective step timestamps. A fixed step length of 0.75 meters is used for all steps. Although simplified, this assumption provides a stable baseline for trajectory reconstruction across different walking sessions.

The full trajectory computation, from preprocessing to plotting, is encapsulated in the `process_pair()` function. This includes data alignment, rotation estimation, heading integration, and step detection. The resulting trajectory is visualized as a 2D plot where each marker represents an individual step, and the line segments illustrate the estimated walking path.

6. Results

Since no ground truth data was available for the recorded trajectories, a quantitative evaluation of the reconstruction accuracy was not feasible. However, a floor plan of the environment was available and used as a reference for qualitative assessment. The reconstructed paths were visually compared against this layout to judge plausibility and consistency. An excerpt of the floor plan is shown in *Figure 7*.

Among the six recorded sessions, track 5 was visually identified as the most consistent and interpretable result and was thus selected as the reference track for the following evaluations. Using this track, we systematically analyzed the influence of different configurations for signal smoothing, peak detection, and step start estimation.

In the first experiment, we varied the noise removal method, while keeping the peak detection algorithm and the step start estimation strategy fixed. After that, we held the chosen noise removal method constant and varied the second parameter, and so on.

Below, we present the smoothing results obtained from the three filtering techniques applied: Butterworth low-pass, Moving Average, and Savitzky-Golay. For all configurations, peak detection was fixed to `find_peaks` from SciPy, and for step start estimation we used the `first` method.

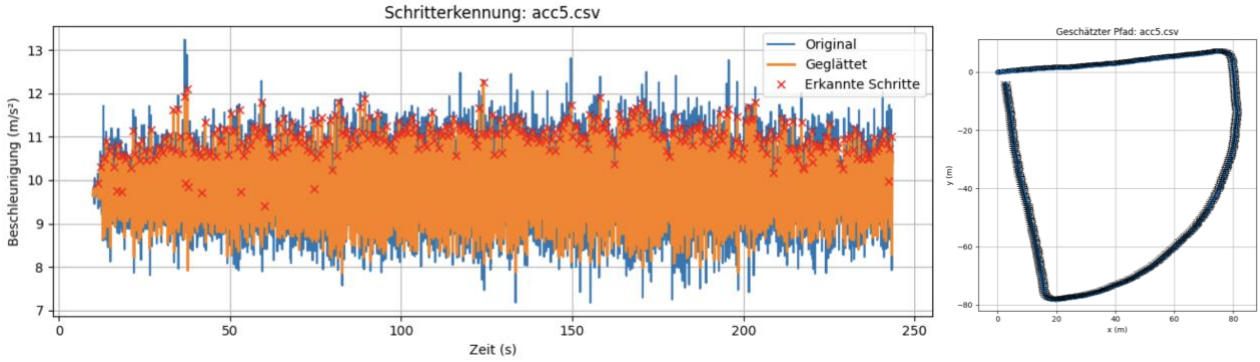

Figure 1: Noise Removal using Butterworth low-pass filter (left) and estimated track (right)

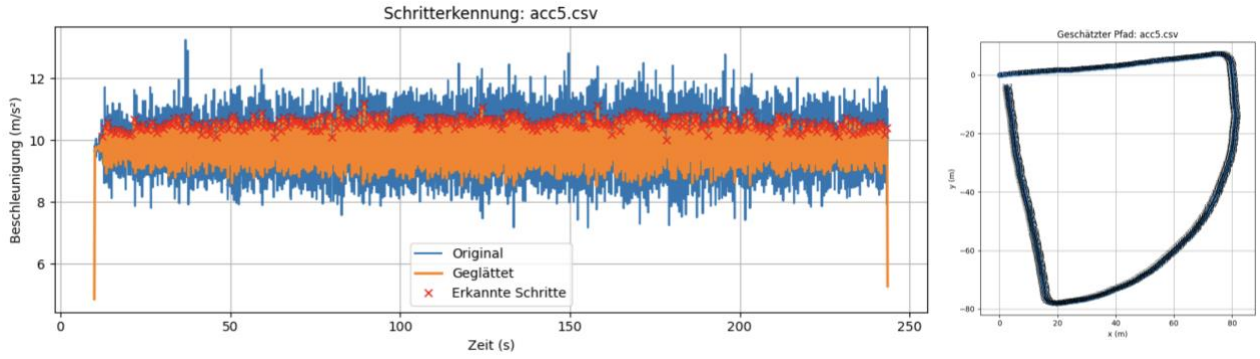

Figure 2: Noise Removal using moving average (left) and estimated track (right)

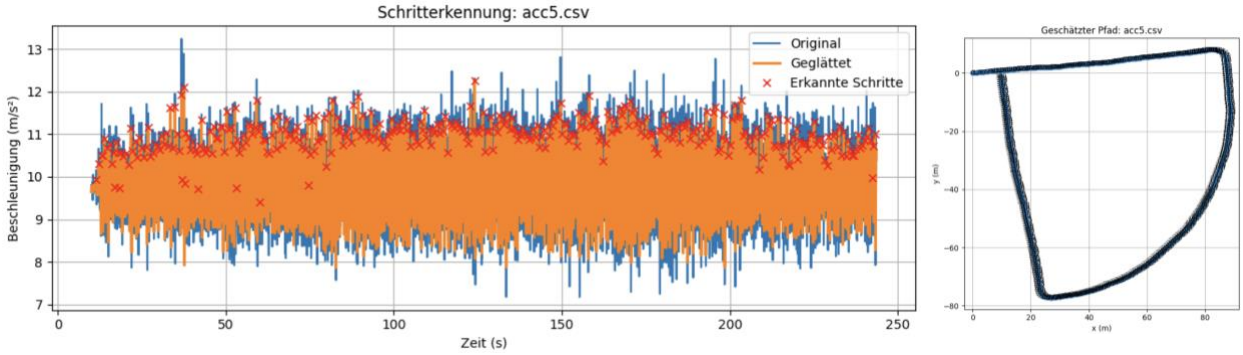

Figure 3: Noise Removal using Savitzky-Golay (left) and estimated track (right)

The number of detected steps and the estimated step start time for each filter configuration are summarized in *Table 1*.

	Butterworth low-pass	**Moving average**	**Savitzky-Golay**
Detected steps	376	376	380
Estimated step start	12.24 s	12.31 s	11.47 s

Table 1: Detected steps and Estimated step starts for each filter

The results indicate that all filters, given the selected parameters, produce comparable outcomes in terms of step detection performance.

For the following experiments on peak detection algorithms and step start estimation, the noise removal method was fixed to the Butterworth low-pass filter.

When using the smoothed z-score method for peak detection, the results became difficult to interpret, and the estimated trajectory deviated significantly from the expected path, as illustrated in *Figure 4*.

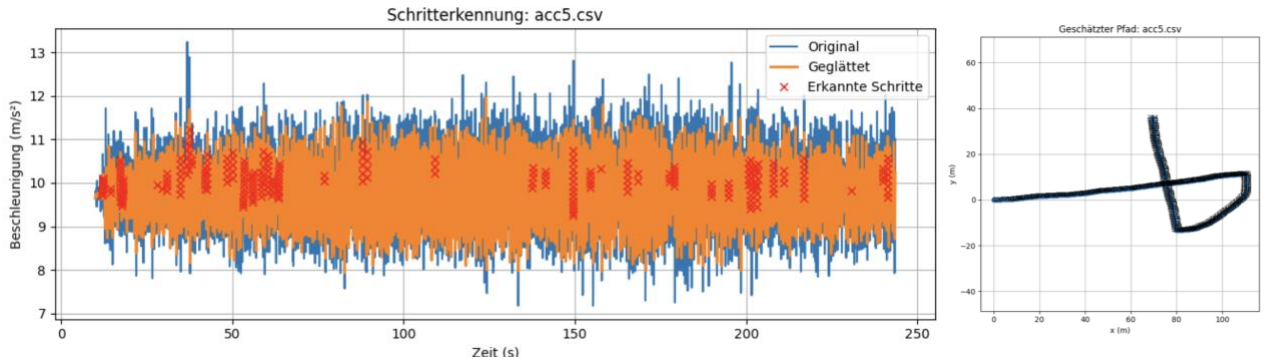

Figure 4: Peak detection using smoothed z-score (left) and estimated track (right)

Unlike the `find_peaks()` function, which explicitly locates local maxima in the signal (see *Figures 1-3*), the `smoothed_z_score()` method is not a peak localization algorithm, but rather an anomaly detection technique. It flags time points where the signal exceeds a dynamically computed threshold based on a moving average and standard deviation.

Consequently, the detected "peaks" do not necessarily coincide with the actual local maxima of the acceleration signal, which can lead to misaligned or spurious step detections. The z-score-based approach is highly sensitive to noise and strongly dependent on its parameter settings (lag, threshold, influence). In practice, this sensitivity becomes especially problematic in cases of non-uniform walking patterns (e.g., brief stops or turns), where the algorithm may either miss valid steps or falsely detect peaks.

This sensitivity directly affects trajectory estimation. As illustrated in *Figure 4 (right)*, the use of z-score-based peak detection led to a significant distortion in the reconstructed path. Some reasons for this deviation might be faulty step detections, inaccurate step timing, or temporal misalignment.

These issues highlight the need for careful parameter tuning when using statistical detection methods like smoothed z-score, and suggest that classic peak-based approaches may yield more robust results for structured gait patterns.

Keeping the Butterworth low-pass filter as the noise removal method and selecting `find_peaks` from SciPy for peak detection, since it produced more reliable results than the smoothed z-score method, we now vary the algorithm used for step start estimation.

This variation does not significantly influence the trajectory reconstruction, and therefore yields very similar outcomes, as reflected in *Figures 1 to 3*, both in terms of detected steps and the estimated paths.

These findings suggest that even a simple method, such as taking the timestamp of the first detected peak, can produce robust and accurate results, comparable to more sophisticated approaches like averaging consecutive peak timestamps or using the minimum between peaks.

As the differences were minimal, we chose to not include additional plots for this part.

Before reconstructing the final path, the user's heading angle (yaw) was estimated over the entire recording by integrating the gyroscope data. *Figure 5* illustrates the evolution of this angle over time for track 5. The plot clearly reflects the user's turning movements, with distinct changes visible around the 75-second and 180-second marks, corresponding to sharp directional turns.

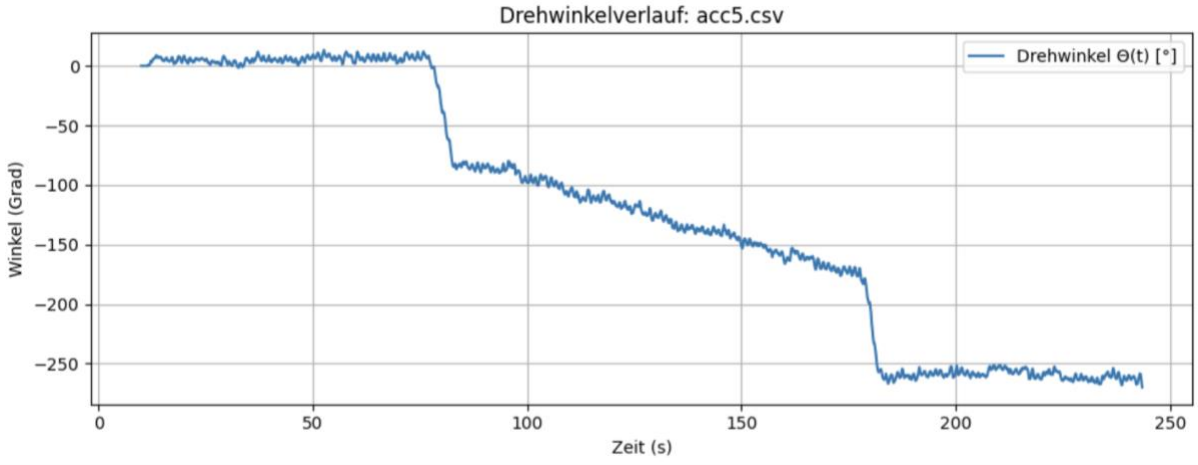

Figure 5: Turning angle profile over time

The final output of the PDR pipeline is the reconstructed 2D trajectory of the user. This path is synthesized by sequentially combining the detected steps with their corresponding heading angles derived from the gyroscope. Each step advances the position by a fixed length of 0.75 meters in the estimated direction.

Figure 6 displays the reconstructed trajectory for track 5, compared to the "ground truth" floor plan, illustrated in *Figure 7*. The path clearly illustrates the system's ability to translate raw inertial data into a coherent spatial representation, capturing both straight walking segments and turns.

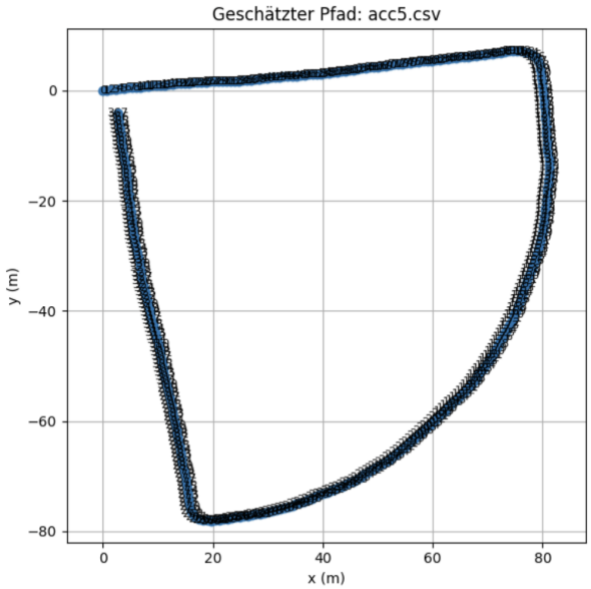

Figure 6: Estimated trajectory

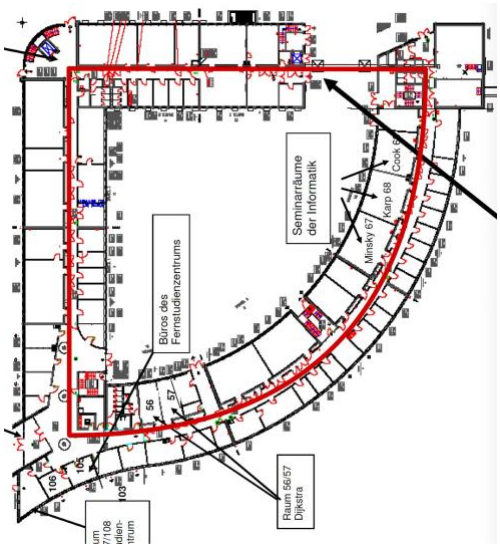

Figure 7: "Ground truth" Floor Plan

7. Conclusion

This project successfully demonstrated the implementation of a complete Pedestrian Dead Reckoning (PDR) pipeline for indoor localization using only inertial data from a smartphone. The system reliably converts raw accelerometer and gyroscope measurements into a coherent 2D trajectory through a sequence of processing steps including signal smoothing, step detection, and heading estimation via gyroscope integration.

The evaluation, based on qualitative comparison with a floor plan (*Figure 7*), shows that the estimated trajectory (*Figure 6*) closely reflects the actual path, including straight segments and directional turns. Experiments with different configurations for signal filtering, peak detection, and step start estimation (*Figures 1–4*) confirmed that even simple algorithmic choices can yield robust results, as long as parameters are chosen appropriately.

At the same time, the results highlight typical limitations of a deterministic PDR system. In particular, drift in the heading angle and the use of a fixed step length introduce cumulative errors over time. These effects are visible in the reconstructed paths and are inherent challenges when no external corrections (e.g., landmarks or map constraints) are applied.

8. Structure of the Python Code

The structure of our code, along with some usage instructions, can be found in the README file.

9. References

[1] Staacks, S., Hütz, S., Heinke, H., & Stampfer, C. (2018). Advanced tools for smartphone-based experiments: phyphox. *Physics Education, 53*(4), 045009.
[2] A. Savitzky und M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least-Squares Procedures," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627-1639, 1964, doi:10.1021/ac60214a047.