A BAYESIAN OCCUPANCY GRID FILTER FOR ROBUST PEDESTRIAN DEAD RECKONING

BY

SAHIL BHANDARY KARNOOR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Adviser:

Professor Romit Roy Choudhury

Abstract

This thesis considers the problem of indoor localization using inertial sensors (IMUs) that are embedded in almost all smartphones and mobile devices. Significant research has focused on developing pedestrian dead reckoning (PDR) algorithms that utilize the IMU measurements to track human movement. While Particle Filters (PF) have offered the best-known accuracy so far, they are also known to suffer from low robustness. This is not surprising given how IMU data from the real world is highly noisy, mainly due to the arm and limb gestures of the user.

Since real-world deployments often favor robustness over accuracy, I propose a Bayesian Occupancy Grid Filter (BOF) that can absorb far greater IMU error compared to PFs. The robustness gains are shown through extensive simulations and the implementation of a fully functional real-time system that performs in accordance with our expectations. BOFs are also simple to implement and can be an important step toward the wide-scale deployment of IMU-based indoor positioning systems.

To Pappa and Amma.

Acknowledgments

I extend my deepest gratitude to the University of Illinois at Urbana-Champaign (UIUC) for providing me with an enriching academic environment and resources essential for the completion of this thesis. The support and guidance offered by the faculty, staff, and fellow researchers at UIUC have been invaluable throughout my academic journey.

I am profoundly grateful to my advisor, Romit Roy Choudhury, for his unwavering support, invaluable insights, and continuous encouragement. His mentorship has been instrumental in shaping my research and fostering my growth as a scholar. His dedication and commitment to excellence have been a constant source of inspiration.

I would like to express my heartfelt appreciation to my collaborators whose contributions significantly enriched this work. Their expertise, collaboration, and willingness to share ideas have been instrumental in shaping the outcomes presented in this thesis. I am indebted to Avinash Subramaniam and Eric Dong for their invaluable assistance and collaboration throughout this research endeavor.

Avinash Subramaniam developed the audio interface on the real-time Android platform for navigation. His excellent C++ code was also invaluable in helping me get started with native development on Android to implement the BOF. Eric Dong was responsible for developing the hardware for the smartglass platform. The demo would not have been possible without the two of them.

I am also indebted to Yu-Lin Wei and Zhijian Zang for their eye-opening insights throughout the project. This thesis would not have been possible without our long back-and-forths.

I am also thankful to my friends and family for their unwavering support, understanding, and encouragement during this academic pursuit. Their belief in my abilities has been a driving force behind my accomplishments.

Lastly, I extend my appreciation to all individuals who, directly or indirectly, contributed to this thesis. Your support and guidance have been indispensable and have significantly contributed to the successful completion of this work.

Table of Contents

List of Abbreviations

PDR	Pedestrian Dead Reckoning
IMU	Inertial Measurement Unit
GPS	Global Positioning System
BOF	Bayesian Occupancy Grid Filter
PF	Particle Filter
DTW	Dynamic Time Warping
RSSI	Received Signal Strength Indicator

Chapter 1

Indoor Localization

Imagine a user, Alice, who wants to navigate to a specific destination within a very large office space. Conventionally, this would require her to use maps or direction signs to navigate the space. This is in contrast to outdoor navigation, where a user can use a GPS device to navigate safely to their destination. Despite digital personal devices becoming pervasive, we do not yet have a mass-market deployment of an indoor navigation system with the same maturity in functionality that outdoor navigation systems have provided for decades. One primary challenge holding back the mass deployment is the lack of a localization module that requires minimal setup, either in terms of hardware infrastructure deployment or time.

This thesis focuses on eliminating external hardware infrastructure requirements. Most of the approaches for indoor localization in the literature require dense deployments of hardware systems to estimate the user's position. Common hardware solutions include Bluetooth, Wifi, and Ultrawideband beacons. One could argue that many interior spaces today have the required hardware, such as WiFi routers, to provide seamless networking. However, these routers may require modification since localization necessitates access to the physical channel [1], [2], which is not universally available. A mass-market system should thus avoid the need for any hardware infrastructure. Furthermore, RF systems require continuous scanning of wireless channels that consume large amounts of power.

The only other modalities left are cameras and Inertial Measurement Units(IMUs) because of the restriction that only sensors built into personal devices can be used. Visual odometry is a mature field that offers algorithms that perform remarkably well in localization applications. Unfortunately, they come with a caveat: the lack of privacy. Having a camera recording continuously within a space may not be possible in secure environments in modern office spaces. We are thus left with just the built-in IMU of personal devices to perform localization. The IMUs of personal devices have the added benefit of low energy requirements.

IMUs are typically used to estimate device orientation. Beyond this, they have found limited utility in estimating the motion activity. Although the acceleration measurements provided by the accelerometer can theoretically be double integrated over time to track position over time, as shown in Figure 1.2, error grows quadratically due to noise. To address this, systems have relied on constraints inherent to the type of motion they track. IMUs in smartwatches attached to the wrist have been used to track hand movements [3] by placing constraints on the range of motion and wrist orientation. When using a gyroscope to estimate wrist orientation, we can restrict the range of motion possible when double integrating accelerometer measurements.

Similar constraints can be exploited on the range of motion possible for localizing a human user within an environment. More specifically, as a person moves along a certain trajectory, we can determine when a step

has been taken. This is done by looking for specific signatures in the IMU signals corresponding to when a step is taken. We can then update the position estimate based on their facing direction and step length. This is termed Pedestrian Dead Reckoning(PDR) [4], [5]. However, this problem is fundamentally hard. The IMU of a personal device is severely affected by activities of the limbs, such as motions of the arms and legs, typing, talking, etc. The IMU senses the "sum" of all these motions and accurately separates the body's displacement from the limb's interference has almost been impossible.

Furthermore, the signatures may not be unique to a human's step. Other kinds of motion, such as jumping or shaking, may also result in similar signatures. It is paramount that we distinguish between these signatures reliably for accurate localization. Another issue is that step lengths are not fixed quantities. PDR must be robust against uncertain step lengths.

Dead reckoning, a term borrowed from oceanic navigation, involves accumulating position change measurements over time. Since it doesn't use any absolute reference for determining position, errors in position change measurement accumulate over time, causing the position estimate to drift from the true position. To correct these errors, localization systems have used sensor signal signatures to find "landmarks" that serve as absolute references to correct errors [6]. But these systems require setup in the form of location-specific datasets [7], [8] collected by repeated traversals throughout the indoor environment, tagged with ground truth data. This data must then be processed offline to determine location-tagged signatures that can be consistently determined. Finally, data corresponding to these signatures must be installed into the application whenever the user changes location. This introduced overhead may restrict the deployment of the indoor localization system in real-world environments.

I restrict the system to localization using a pedestrian dead reckoning algorithm to remove the need for the aforementioned requirements. PDR is used in a probabilistic framework that restricts the probability space to within the freely traversable area of the indoor environment. This needs a floorplan of the indoor space, a requirement that is met because of the navigation component of the system. This map serves as a "filter," constraining the position estimates to within the environment that a person can feasibly reach indoors.

A common approach to tackling this problem is the particle filter(PF). PFs are initialized with a set of particles placed at candidate user locations (a prior). As the user traverses the environment (i.e., the IMU has detected a new step), all the particles are updated based on the IMU measurements and the user's motion model. When a particle crosses a barrier in the map, the particle is eliminated (since it could not have been a correct estimate of the user's location), and a new particle is spawned (or "resampled") at one of the locations of the left-over particles. As a result, the number of particles remains the same, and over time, they are expected to converge around the user's actual position.

PFs are designed to absorb some uncertainty because each particle can track one of many possible location estimates. The larger the number of particles, the greater the PF's ability to absorb uncertainty. However, the particle filter can diverge beyond a threshold uncertainty, meaning that none of the particles are within some radius of the true user location. Figure 1.1 shows a simple example where, at time t_0, the particles are all initialized correctly at the user's location. If the user's IMU detects a few false steps, all the particles can get propagated to incorrect locations along a wall; if the user now makes a turn, all the particles get eliminated since they walk into the wall. Without a second modality that can measure the user's absolute location, there is no principled way to resample the particles.

Towards a more robust probabilistic framework for PDR, I propose a *Bayesian Occupancy Grid Filter* (BOF). This filter divides the indoor floor plan into grid cells. Each grid cell that doesn't contain a wall or an

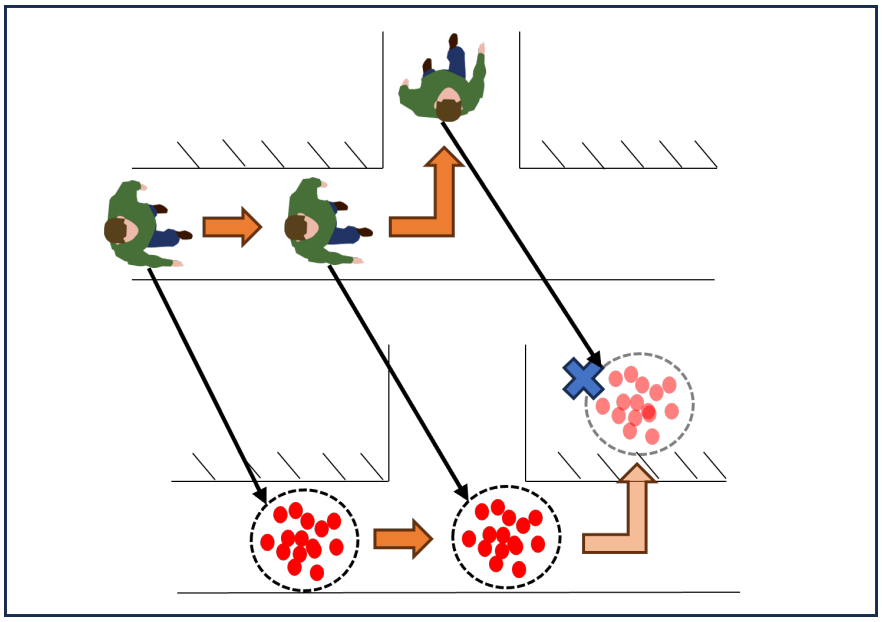


Figure 1.1: Divergence of PF due to False Step Count/Length

obstacle stores the probability that the user is within the cell. These cell probabilities are updated according to a motion model that implicitly accounts for the inaccuracies. By design, the BOF can place support at all locations a user can traverse within the indoor environment, allowing it to tolerate far greater uncertainty in step detection, step length, and walking direction. For example, to ensure robustness against false step detections, some probability mass can be retained in the current cell even if the IMU detects a step. This ensures that support remains at the true location despite various uncertainties. The trade-offs are that the size of each cell determines the localization granularity, and the number of cells may get quite large as the indoor map grows. Results show that simple pruning on the grids can mitigate this problem while upholding robustness gains.

Firstly, the PF and BOF are simulated on real floorplans by injecting errors in step count, step length, and walking direction. To maintain parity in computational complexity, all the experiments equalize the number of particles to the number of grid cells in BOF. After uniformly initializing both filters within the indoor environment, convergence is tested after a user has walked for N steps in the building. Results show that PFs handle up to 5% false steps and routinely diverge after that, while BOF can absorb even 28% step errors. Similar gaps exist for erroneous step length and walking direction as well, suggesting a consistent robustness gain from BOF. Of course, the PF achieves higher localization accuracy when they do converge, but when one considers the overall accuracy distribution, it appears that the tradeoff is clearly in favor of BOF.

Finally, a real-time platform is developed to demonstrate BOF in the real world, consisting of (1) a real-time smartphone application and (2) miniature IMU hardware attached to a pair of wearable glasses. Users wear glasses, carry their phones in any way they like, and walk around in our building. Using this platform, the BOF is shown to run in real-time and localizes the user with a high degree of robustness compared to the PF. Our demo video is available at https://sinrg.csl.illinois.edu/.

In summary, the contributions of this thesis are as follows:

- I propose a Bayesian Occupancy Grid Filter (BOF) that is simple to realize and offers greater robustness

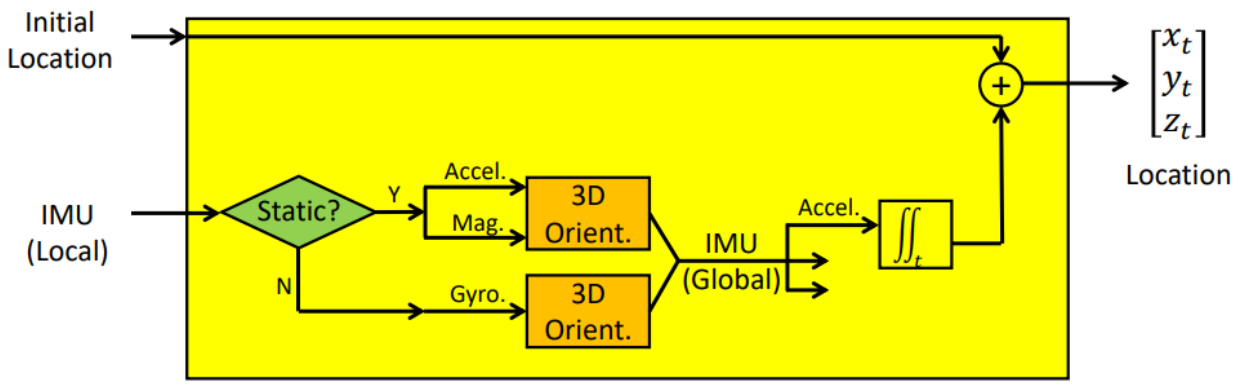


Figure 1.2: Conventional IMU based Dead Reckoning

in PDR-based indoor localization.

- I present simulation results comparing the robustness of the BOF against a Particle Filter (PF) for varying uncertainty parameters.
- I also demo a real-time system using a smartphone and wearable IMU, where the results align with the simulations.

Chapter 2

Pedestrian Dead Reckoning

Pedestrian Dead Reckoning(PDR) is an indoor localization technique that exploits the user's gait to determine their position. Its main appeal is it requires just an IMU attached to their person. Unlike other indoor localization techniques that require additional infrastructure, using just an IMU enables the algorithm to be more widely deployable. With the increasing prevalence of IMUs built into personal devices, a user isn't restricted by the type or form factor of devices they need to localize themselves.

The crux of PDR is detecting when a user takes a step and then updating the estimate of their position by adding a displacement along their facing direction. It first takes an input as an approximate estimate of the user's initial position. Assuming there exists a good estimate of the distance a user covers after every step, the algorithm can track the user reasonably well over a few steps. This chapter covers how both step detection, as well as facing direction, can be tracked using an IMU.

2.1 Inertial Measurement Units

Inertial measurement units (IMUs) consist of 3 odometric sensors: the accelerometer, gyroscope, and magnetometer. Each measures specific quantities along the three standard X, Y, Z axes: the accelerometer measures acceleration, the gyroscope measures angular velocity, and the magnetometer measures the magnetic field.

IMUs are extensively used as a sensing modality in personal devices. Nearly every consumer portable device is outfitted with IMUs to allow them to perform functions such as orientation estimation, motion detection, and activity tracking. Since they sense quantities intrinsically from the device without requiring external infrastructure, they are ubiquitous for localization in both indoor and outdoor scenarios.

As stated earlier, a wide variety of algorithms exist for device orientation estimation using IMUs. The orientation estimates allow us to model the user's facing direction while they traverse the indoor environment. In the following section, we mathematically describe how this is performed.

2.2 Orientation Estimation using Inertial Measurement Units

The three sensors in the IMU can be used to estimate the orientation of a device with respect to a fixed global frame of reference with remarkable accuracy. The gyroscope tracks the angular velocity in the device's frame of reference, which can be used to estimate the global orientation over time by integration. The gyroscope's

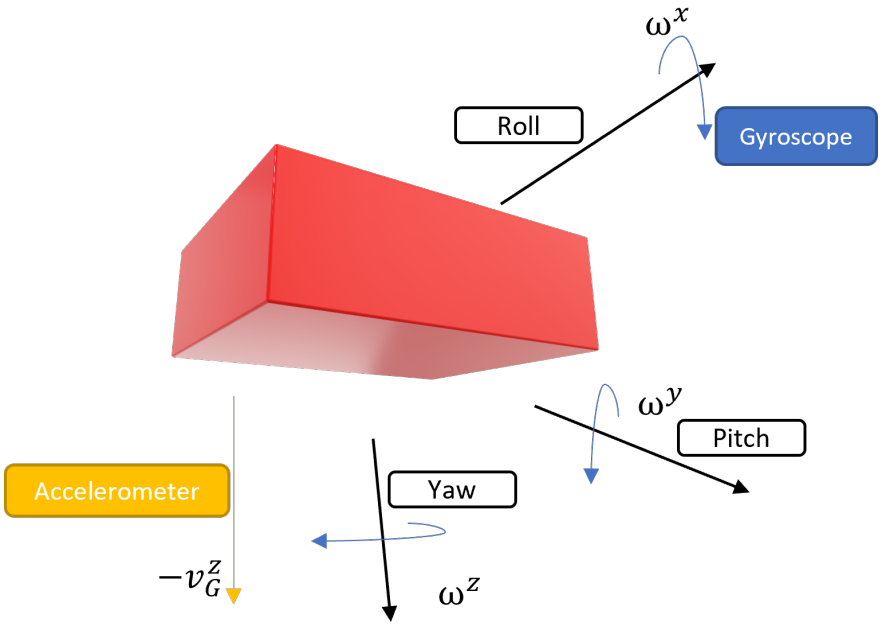


Figure 2.1: Euler angles

measurements, unfortunately, have noise, which can be corrected by incorporating the accelerometer and magnetometer measurements. The accelerometer provides estimates of the gravity vector that can be used to correct errors in the downward axis estimates from the gyroscope. Similarly, the magnetometer measures the Earth's magnetic field. This vector is linearly independent of the gravity vector and can be used to correct errors in the facing direction or heading estimates. The magnetometer can, unfortunately, be used only outdoors because of the presence of magnetic anomalies indoors. These anomalies corrupt the Earth's magnetic field, causing drift in the heading estimates over time.

Orientation of a device can be tracked using three orthonormal vectors $[v_t^x \quad v_t^y \quad v_t^z]$, $v_t^* \in \mathbb{R}^3$ that track the orientation at instant t. These represent the standard X, Y, Z axes of the global frame in the device's local frame of reference, where X points right, Y points forward, and Z points upward. In the local frame, the algorithm assumes that the three standard axes $[e^x \quad e^y \quad e^z]$ correspond to the front, right, and upward directions of the device.

$$e^x = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \; e^y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, \; e^z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{2.1}$$

Since v_t^* are orthonormal vectors, $R_t^L = [v_t^x \quad v_t^y \quad v_t^z]$ represents a rotation matrix that transforms any vector u_{global} in the global frame to the device's local frame.

$$u_{local} = R_t^L \cdot u_{global} \tag{2.2}$$

To track the orientation of the device over time, the algorithm must estimate the matrix R_t^L over instants t. The rotation matrix R_t^L belongs to the rotation group $SO(3)$ which represents rotations in $\mathbb{R}^3$ [9]. This means that given two of its columns, the third can be estimated using the 3D cross product. In practice, the

forward vector v_t^y and the downward vector $-v_t^z$ are tracked. The right vector v_t^x can be computed as:

$$v_t^x = v_t^y \times v_t^z \tag{2.3}$$

Let's assume the algorithm is provided initial estimates of the two vectors v_0^y and $-v_0^z$. These two are tracked over time using angular velocity estimates from the gyroscope. The gyroscope provides three angular velocity measurements. These measurements, denoted by $[\omega_t^x \quad \omega_t^y \quad \omega_t^z]^T$ are provided along the standard axis of the device in its local frame. For example, ω_t^x is the angular velocity about e^x. These measurements can be integrated over time to obtain the change in orientation about the three standard axes:

$$[\theta_t^x \quad \theta_t^y \quad \theta_t^z]^T = \delta t[\omega_t^x \quad \omega_t^y \quad \omega_t^z]^T \tag{2.4}$$

Here δt represents the sampling interval of the gyroscope. The orientation of the device can then be estimated by updating the local estimate of the global axes. The estimate of the global axis changes in the opposite direction as the measured angular change. The following rotation matrices can be defined as follows since the rotation angles are measured about the standard axes:

$$R^x(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(-\theta_t^x) & -sin(-\theta_t^x) \\ 0 & sin(-\theta_t^x) & cos(-\theta_t^x) \end{bmatrix} \tag{2.5}$$

$$R^y(t) = \begin{bmatrix} cos(-\theta_t^y) & 0 & sin(-\theta_t^y) \\ 0 & 1 & 0 \\ -sin(-\theta_t^y) & 0 & cos(-\theta_t^y) \end{bmatrix} \tag{2.6}$$

$$R^z(t) = \begin{bmatrix} cos(-\theta_t^z) & -sin(-\theta_t^z) & 0 \\ sin(-\theta_t^z) & cos(-\theta_t^z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$R(t) = R^x(t) \cdot R^y(t) \cdot R^z(t) \tag{2.8}$$

Thus, the updated estimate of global axes can be computed as:

$$v_{t+\delta t}^y = R(t) \cdot v_t^y \tag{2.9}$$

$$-v_{t+\delta t}^z = R(t) \cdot -v_t^z \tag{2.10}$$

2.2.1 Drift correction using Complementary filtering

Gyroscope measurements tend to be corrupted by noise, which accumulates over time. The orientation estimates could drift quite quickly if this noise isn't corrected. An absolute reference is needed to correct these errors. When the IMU is stationary, the accelerometer provides a noisy estimate $\hat{a}_G^z$ of the vector corresponding to the acceleration due to gravity a_G^z. This vector approximately aligns with the global downward vector $-v^z$ in the local frame of reference. The normalized vector that the accelerometer provides

is denoted with:

$$v_t^G = \frac{\hat{a}_G^z}{||\hat{a}_G^z||} \tag{2.11}$$

The accelerometer estimate of the gravity vector is heavily affected by noise due to the motion of the IMU. Although the mean of the accelerometer's downward direction estimate converges to the true downward direction v^z, this estimate has a high variance. On the other hand, the measurement of the downward direction obtained by integrating the gyroscope measurements $-v_t^z$ has low variance but diverges in mean over time. To get better estimates, both sensors need to be combined.

Various algorithms that fuse both the accelerometer and gyroscope estimates have been proposed. One such algorithm is the complementary filter [9]. In this thesis, this sensor fusion algorithm is implemented.

The complementary filter performs a weighted average on two sensor measurements. The sensor with high-frequency noise is provided with a smaller weighing factor, while the sensor with better low-frequency performance is provided with a larger weight factor. In the case of fusing the accelerometer and the gyroscope, the gyroscope-based measurements have low variance but drift over time. This can be thought of as low-frequency noise. On the other hand, the accelerometer converges to the true value in mean but has a lot of variance in its measurements. This can be characterized as high-frequency noise. Both sensors can be thought of as behaving in a complementary manner.

Let's denote the mixing coefficients as α, β. Since the fused measurements are to be an unbiased estimate of the quantity, they must sum to 1.

$$\begin{aligned} \alpha > 0,\ \beta > 0 \\ \alpha + \beta = 1 \end{aligned} \tag{2.12}$$

The fused estimate of the downward vector $-\hat{v}_t^z$ can now be calculated after normalizing:

$$-\hat{v}_t^z = \frac{\alpha(-v_t^z) + \beta v_t^G}{||\alpha(-v_t^z) + \beta v_t^G||} \tag{2.13}$$

This new estimate will no longer be orthonormal to the forward vector v_t^y, although it should still be linearly independent of it. The two vectors can now be orthonormalized to obtain our optimal estimates of the forward and downward vectors. Since $-\hat{v}_t^z$ is our current optimal estimate of the downward direction, this vector can be preserved, and instead, the orthonormal vector that spans the same subspace as v_t^y and itself can be found.

This can be performed using cross-products. The subspace spanned by the downward and forward vector is orthogonal to the right vector. Thus, the optimal forward vector $\hat{v}_t^y$ can be computed by performing two consecutive inner products.

$$\hat{v}_t^x = \frac{v_t^y \times \hat{v}_t^z}{||v_t^y \times \hat{v}_t^z||} \tag{2.14}$$

$$\hat{v}_t^y = \hat{v}_t^z \times \hat{v}_t^x \tag{2.15}$$

In outdoor environments, the magnetometer provides a similar reference for the forward vector v_t^y in the local frame by measuring the Earth's magnetic field. Unfortunately, the magnetometer cannot be utilized indoors due to the presence of magnetic anomalies. This results in an error persisting while estimating v_t^y, and consequently, v_t^x, which causes yaw estimates to drift from the true value over time.

2.2.2 Heading Estimation

Using $\hat{v}_t^x, \hat{v}_t^y$ and $\hat{v}_t^z$, the heading Θ_t of the device can be estimated. First, the rotation matrix $\hat{R}_t^L$ can be expressed as $\hat{R}_t^L = [\hat{v}_t^x \quad \hat{v}_t^y \quad \hat{v}_t^y]$. This matrix can be decomposed into Euler angles Θ, Φ, and Ψ. Θ is the angle about the Z axis and is the heading. Φ is the angle about the Y axis and is the roll, and Ψ is the angle about the X axis and is the pitch. When applied to a global vector, $\hat{R}_t^L$ performs rotations based on these Euler angles to transform the vector into the IMU's local coordinates, 2.2. To perform this rotation, $\hat{R}_t^L$ applies the inverse of these angles to obtain the local vector. Lets denote this inverse as $\hat{R}_t^G$. Since $\hat{R}_t^L$ is an orthonormal matrix, its inverse is its transpose.

$$\hat{R}_t^G = (\hat{R}_t^L)^T \tag{2.16}$$

$$u_{global} = R_t^G \cdot u_{local} \tag{2.17}$$

This matrix is decomposed into the product of the individual rotation matrices described by the Euler angles:

$$\hat{R}_t^G = \hat{R}_t^x(\Psi) \cdot \hat{R}_t^y(\Phi) \cdot \hat{R}_t^z(\Theta) \tag{2.18}$$

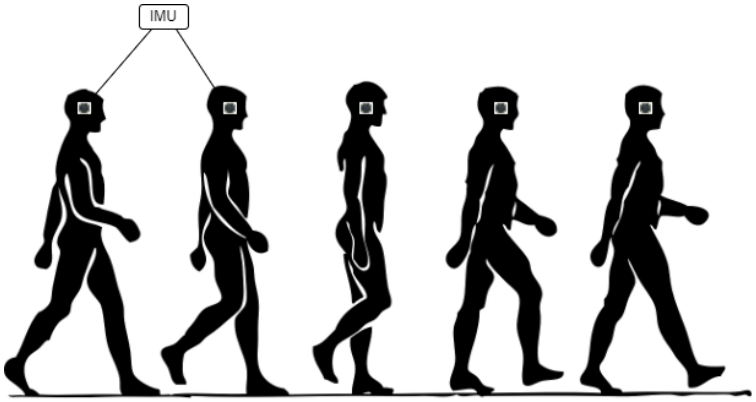
$$= \begin{bmatrix} cos(\Psi)cos(\Theta) & -cos(\Psi)sin(\Theta) & sin(\Psi) \\ cos(\Psi)sin(\Theta) + sin(\Psi)sin(\Phi)cos(\Theta) & cos(\Psi)cos(\Theta) - sin(\Psi)sin(\Phi)sin(\Theta) & -sin(\Psi)cos(\Phi) \\ sin(\Psi)sin(\Theta) - cos(\Psi)sin(\Phi)cos(\Theta) & sin(\Psi)cos(\Theta) + cos(\Psi)sin(\Phi)sin(\Theta) & cos(\Psi)cos(\Phi) \end{bmatrix} \tag{2.19}$$

Thus, the heading of the IMU at instant t, Θ_t is given by the following, which is assumed to correspond to the user's facing direction.

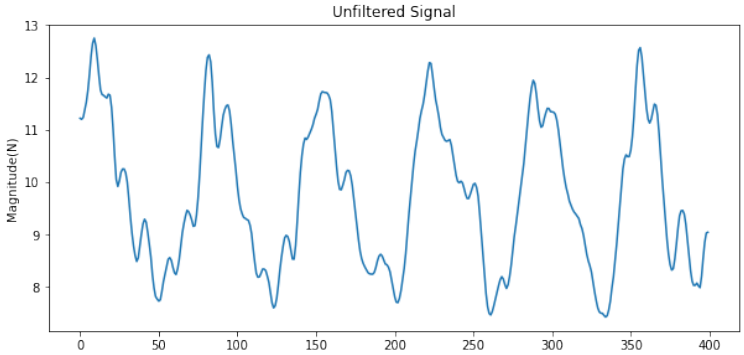
$$\Theta_t = tan^{-1}(-\frac{\hat{R}_t^G[1,2]}{\hat{R}_t^G[1,1]}) \tag{2.20}$$

2.3 Step Detection

When a person takes a step, the accelerometer picks up this signal as a distinct signature, as shown in figure 2.2b. The figure shows the total magnitude of the accelerometer across the three orthogonal axes. A reference template can be defined corresponding to this signal, and this template can be searched in the accelerometer input stream. Since the duration of a step may vary based on factors such as the person's speed, this signature may appear warped along the time axis. DTW distance is used to find the step signature in the accelerometer signal.



(a) Human Locomotion



(b) Accelerometer signal

Figure 2.2: Step Detection

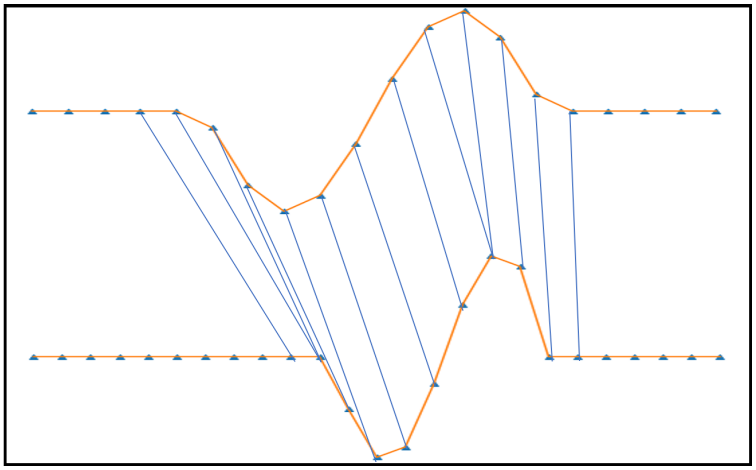


Figure 2.3: Template matching using Dynamic Time Warping

2.3.1 Dynamic Time Warping

When two sequences need to be compared, a common metric is to use an L2 norm after aligning them. If there are two vectors $X, Y \in \mathbb{R}^n$, the distance between them can be defined as

$$d(X, Y) = ||X - Y||_2.$$

In the time series template matching problem, a common scenario is having one of the sequences warped along the time axis. Thus, both sequences may no longer be of the same length. Dynamic Time Warping(DTW) is an algorithm that provides a metric to compare two sequences that are invariant under warping along the time axis. This is done by allowing many-to-one mappings between the two sequences. Given two sequences $X = (x_1, ..xo) \in \mathbb{R}^o$ and $Y = (y_1, ..y_p) \in \mathbb{R}^p$, DTW$(X, Y)$ is defined as:

$$\begin{aligned}
&\text{DTW}(X, Y) = f(o, p) \\
&f(i, j) = ||x_i - y_j||_2 + min \begin{cases} f(i-1, j-1) \\ f(i-1, j) \\ f(i, j-1) \end{cases} \\
&f(0, 0) = 0
\end{aligned}$$

This distance can be efficiently computed using dynamic programming. For matching a data stream to a fixed template, there exists a modification to this algorithm called SPRING [10].

2.3.2 Variable Step Type

An important factor to consider is the variation in the type of stride taken by the user. The signature observed in the accelerometer signal will differ based on the striding speed, which is low when walking slowly, in contrast to running. Unique templates can be defined for each type of movement to distinguish between the types of movement in a multiple-hypothesis testing framework. Let $S^k \in \mathbb{R}^o, k \in \{0, ..K\}$ be the K unique templates corresponding to the unique stride sequences. The stream of indexed accelerometer samples

$\{a_q\}, q \in \mathbb{N}$ come in at regular time intervals, with a_s being the latest sample. Let $a_r, r < s$ refer to the last accelerometer sample of the previously detected step. The DTW distance can be found between observed sequence $A = \{a_{r+1}, ..a_s\}$ and $\{S^k\}$:

$$d_k(A) = \text{DTW}(A, S^k).$$

The hypothesis can be referred to as Y^k corresponding to the type of stride k, with Y^0 being the null hypothesis. Its likelihood can be defined as $P_k(A) \sim p_k(d_k(A))$. The priors can be defined as π_k based on past step information and assume uniform costs. The Bayes decision rule $\delta_B(A)$ is expressed by:

$$\delta_B(A) = \underset{0 \leq k \leq K}{\text{argmax}}\ \pi_k p_k(d_k(A))$$

2.3.3 Uncertainty in Step Detection and Length

A variance in the step length always exists, even when tailor-made for a particular user. Regardless of the number of IMU signal templates defined for different types of steps, the PDR algorithm needs to account for variances in the step length. Furthermore, there is always a finite probability of a false step detection. A template for a step might be spuriously detected due to the irregular motion of a user. PDR needs to be modeled as a probabilistic estimation problem to account for these uncertainties.

Chapter 3

Particle Filter for Pedestrian Dead Reckoning

Pedestrian dead reckoning can be modeled as a linear dynamic system. This state-space system is defined as a random variable that evolves over time. Assuming the position of the user is $\overline{X}_{n-1}$ after $n-1$ steps, the user's position transitions as follows whenever a step is detected:

$$\overline{X}_{n-1} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} + Y_n^k \begin{bmatrix} cos(\Theta_n) \\ sin(\Theta_n) \end{bmatrix} \tag{3.2}$$

Here, position evolves in 2D Euclidean space, with coordinates x_n, y_n. The step length corresponds to Y_n^k, and the user's facing direction at the current step is denoted by Θ_n. This model assumes free space, i.e., no map constraints are enforced. This dynamic model can be termed as $q(\overline{x}_n | x_{n-1}, Y_n^k, \Theta_n)$.

3.1 Recursive Filtering

Because of imperfect information, position must be modeled as a random variable. The current position estimate $\overline{X}_n$ where $\overline{x}_n = [x_n \quad y_n]^T \in \mathbb{R}^2$ has a probability density $P(\overline{x}_n | \Theta_{1:n}, Y_{1:n}^k, M)$, where M refers to the constraints placed by the indoor map. This density can be modeled recursively based on the sequence of step lengths, facing directions, and map constraints.

$$P(\overline{x}_n | \Theta_{1:n}, Y_{1:n}^k, M) = \int_{\mathbb{R}^2} P(\overline{x}_n | \overline{x}_{n-1}, \Theta_n, Y_n^k, M) P(\overline{x}_{n-1} | \Theta_{1:n-1}, Y_{1:n-1}^k, M) d\overline{x}_{n-1} \tag{3.3}$$

The term $P(\overline{x}_n | \overline{x}_{n-1}, \Theta_n, Y_n^k, M)$ represents the dynamic model, while $P(\overline{x}_{n-1} | \Theta_{1:n-1}, Y_{1:n-1}^k, M)$ represents the position distribution at the previous step. A Kalman filter can estimate this system since the dynamic model is linear and recursive.

Linear models such as the Kalman filter [11] are well known to model linear state-space systems. The

Kalman models the variable of interest as a unimodal Gaussian Distribution that evolves according to the Linear dynamic model. Unfortunately, these linear models fail when the underlying space is discontinuous or the state has to be modeled as a multi-modal probability distribution.

The state space of an indoor environment is no longer a 2D Euclidean space because of the presence of walls and obstacles over which one cannot place any non-zero probability mass. A Kalman Filter cannot be directly used to enforce map constraints since it assumes support over Euclidean or locally Euclidean spaces such as differentiable manifolds [12]. The discontinuities cannot be modeled directly. Some extensions to the Kalman filter that can incorporate map information exist. M. F. Mansour and D. W. Waters [13] designate infeasible regions in the map as polygons and project the state estimate onto the closest polygonal edge. Unfortunately, it is still inherently unable to represent multi-modal distributions.

3.2 Particle Filtering

Particle filtering [14], which is a Sequential Monte Carlo method, is ideal for not only modeling distributions in discontinuous probability spaces, but also for non-linear dynamic systems. The crux of the particle filter is to generate N IID samples corresponding to the required distribution $P(\overline{x}_n|\Theta_{1:n}, Y^k_{1:n}, M)$. It then outputs the conditional mean predictor $\tilde{X}_n$

$$\tilde{X}_n = \frac{1}{N}\sum_u \overline{x}^{[u]}_n \tag{3.4}$$

which is the sample mean of the N IID samples drawn from the required distribution.

The sampling process is performed as follows. The u^{th} particle can be sampled from the position distribution at the previous step as:

$$\overline{x}^{[u]}_{n-1} \sim P(\overline{x}_{n-1}|\Theta_{1:n-1}, Y^k_{1:n-1}, M) \tag{3.5}$$

The particle can then be propagated using $P(\overline{x}^{[u]}_n|\overline{x}_{n-1}, \Theta_n, Y^k_n, M)$. Unfortunately, this dynamic model has no closed-form expression since the map M enforces discontinuities. Instead, the particles are propagated according to the linear dynamic model $q(\overline{x}_n|x_{n-1}, Y^k_n, \Theta_n)$.

$$\overline{x}^{[u]}_n \sim q(.|\overline{x}^{[u]}_{n-1}, Y^k_n, \Theta_n) \tag{3.6}$$

The probability distribution represented by the particles now corresponds to $P(\overline{x}_n|\overline{x}_{n-1}, \Theta_n, Y^k_n)$, which approximates our required distribution. It doesn't incorporate the map information. To get the true distribution, weights are assigned to each of the particles through importance sampling. Each particle is assigned an importance weight based on how well it represents the true probability distribution. These importance weights are the ratio of the true probability distribution that needs to be estimated with respect to the approximated probability distribution.

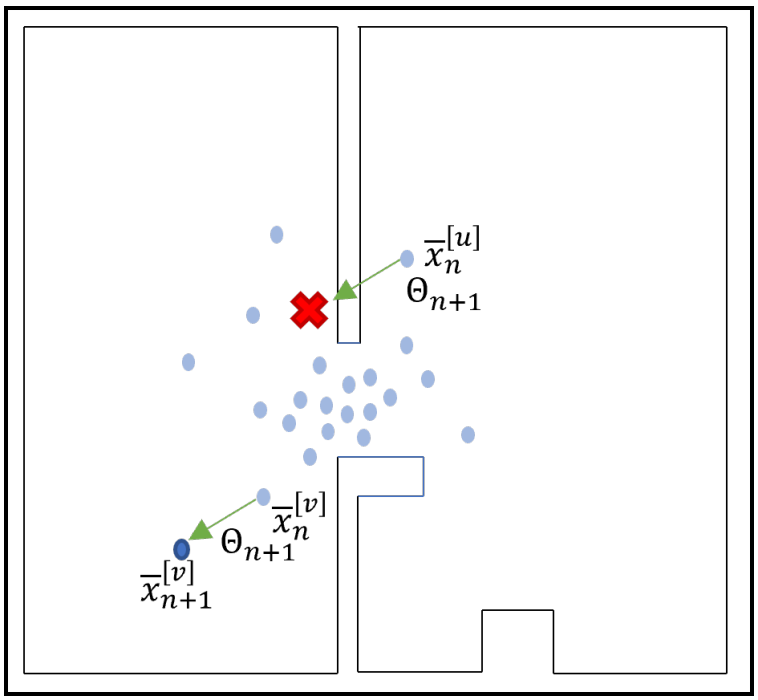


Figure 3.1: Particle Filtering for PDR

$$w^{[u]} = \frac{P(\overline{x}_n^{[u]}|\overline{x}_{n-1}, \Theta_n, Y_n^k, M)}{P(\overline{x}_n^{[u]}|\overline{x}_{n-1}, \Theta_n, Y_n^k)} \tag{3.7}$$

$$= \frac{P(M|\overline{x}_n^{[u]}, \overline{x}_{n-1}, \Theta_n, Y_n^k)}{p(M|\overline{x}_{n-1})} \tag{3.8}$$

$$= \mathbb{1}\{\overline{x}_n^{[u]} \textit{ collides with } M\} \tag{3.9}$$

The indicator function $\mathbb{1}\{\overline{x}_n^{[u]} \textit{ collides with } M\}$ chooses particles that have valid updates given the map constraints. Given the step length and azimuth, particles that collide with or cross barriers are given zero weights, eliminating them, while the ones that survive represent the true distribution.

Surviving particles can now be resampled so as to keep the number of particles constant after successive steps. Over time, the particles should converge to the user's true position as incorrectly initialized particles are filtered out by the map.

3.3 Robustness against Uncertainty

To improve the robustness of the PF, uncertainties in the step length and facing direction must be accounted for. A standard technique for this is sampling the step length and facing direction for every update from a distribution rather than the estimated values. Step lengths and headings can be sampled from distributions, with the step length mean as the detected step Y_n^k's length and the heading mean as Θ_n, respectively. With enough particles, the PF should capture the true distribution of the user's position.

Another uncertainty that needs to be accounted for is during step detection. False step detections can

cause particles at the true locations to propagate incorrectly within the indoor map. These propagated particles may be eliminated because of collisions with a barrier present along the facing direction of the false step. To mitigate this issue, modification is proposed where, for every particle, a binary random variable $Q^{[u]}$ is sampled with probability p when a transition occurs. If the $Q^{[u]}$ turns out to be 1, the particle is propagated according to the transition model, but the particle is kept static if $Q^{[u]}$ equals 0.

Even with the above modification to the Particle Filter, divergence is still observed during Pedestrian Dead Reckoning if the error regimes are large in estimating the facing direction and step parameters. A collapse in the distribution causes this if the map filters out particles representing the true position. In our experiments, this issue persists even when the total number of particles was increased.

Chapter 4

Bayesian Occupancy Grid Filter

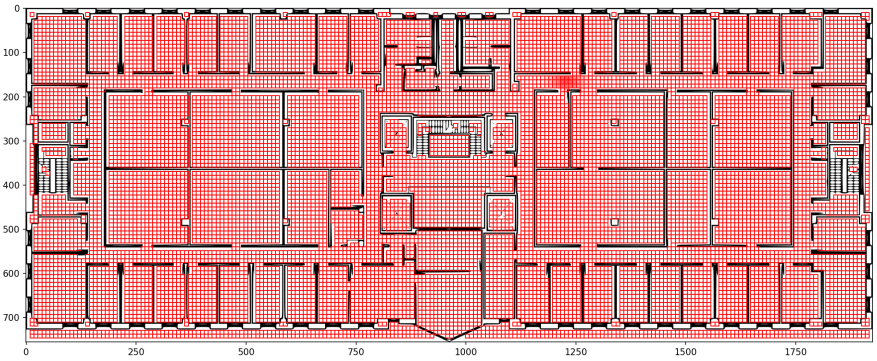
As seen in the previous chapter, a Particle Filter for Pedestrian Dead Reckoning doesn't offer the robustness required to continue tracking a user when there is a high uncertainty step detection, length estimation, and orientation estimation. This occurred primarily due to a lack of sufficient support to represent the user's probability distribution within the indoor environment. When the sensor noise is high, which in this case is the IMU step detection module, particles tend to propagate incorrectly. In the specific case of an indoor environment filled with walls and other obstacles, improperly propagated particles tend to collide with them, resulting in particles being resampled away from areas near the obstacles. If the user's true position is actually near an obstacle, but particles in these areas are propagated incorrectly, the particle filter diverges. Resampling strategies may also fail in certain floor plans without an additional modality to reset the particles to the user's true location.

Since the primary cause of failure was a lack of support in the case of the particle filter, an efficient framework is needed for tracking multi-modal probability densities, a requirement for discontinuous probability spaces, while also placing a finite amount of probability mass at every reachable location. To satisfy this requirement, I propose the Bayesian Occupancy Grid Filter (BOF) that can track such a distribution across a large indoor floor plan. The capability to have finite support at every position in the environment ensures robustness even with only an IMU of a personal device as the sensing modality.

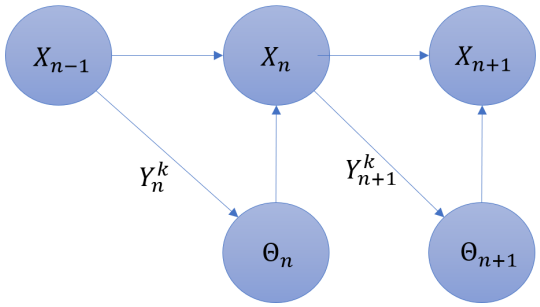
The Bayesian Occupancy Filter (BOF) models the position of the user within an indoor environment after step n, $\overline{X}_n$ using a probabilistic grid. $\overline{X}_n$ is now a random variable with its probability space as a grid superimposed onto the indoor map. Free space is modeled as grid cells c^i, and the absence of a cell represents barriers. A cell c^i is assigned the probability of the presence of the user in that cell after n steps, $P(\overline{X}_n = c^i)$. For brevity, this is denoted as $P(x_n^i)$. This framework can thus capture the multi-modal distribution of the user's position within a discontinuous 2D space.

Representing the probability space as a grid provides finite support at all possible locations a person could occupy within an indoor map. Although the resolution of the grid limits the precision at which we can track a user, this approach prevents the probability collapse that plagues particle filter-based approaches for PDR.

Although the number of grid cells increases by the square of the floor plan length, this does not pose an issue regarding computation in most scenarios for indoor localization. In extremely large floor plans, indoor spaces can be separated into smaller disjoint zones, and the user can be localized within only one of them. Furthermore, most indoor Pedestrian tracking applications require no more than sub-meter level accuracy to provide a good user experience, requiring a precision of no more than a cell length of 30cm. This is far better



(a) Occupancy Grid



(b) Bayesian graph

Figure 4.1: Occupancy Grid Filter

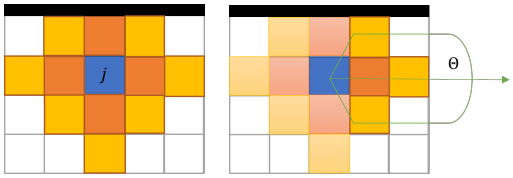
than the 4.9-meter accuracy that GPS provides, even in open-air conditions [15].

4.1 Motion Model

Our goal is to estimate the position $\overline{X}_n$ after n detected steps. This probability evolves as steps $Y_{1:n}^k$ are detected that occur along facing directions $\Theta_{1:n}$, subject to the constraints of map M. The probability density of $\overline{X}_n$, which is modeled by the distribution $P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M)$ is termed the motion model and is represented by the Bayesian Graph in 4.1b. As shown in the Bayesian Graph, the model represents a Markov chain in the position $\overline{X}_n$. The state evolves whenever a step is detected along the estimated facing direction. The motion model can be simplified as follows:

$$
\begin{aligned}
&P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M) \\
&\quad = \frac{P(x_n^i, \Theta_n, Y_n^k|\Theta_{1:n-1}, Y_{1:n-1}^k, M)}{P(\Theta_n, Y_n^k|\Theta_{1:n-1}, Y_{1:n-1}^k, M)} && (4.1)\\
&\quad = \eta P(x_n^i, \Theta_n, Y_n^k|\Theta_{1:n-1}, Y_{1:n-1}^k, M) && (4.2)\\
&\quad = \eta \sum_j P(x_n^i, \Theta_n, Y_n^k|x_n^j, \Theta_{1:n-1}, Y_{1:n-1}^k, M) P(x_{n-1}^j|\Theta_{1:n-1}, Y_{1:n-1}^k, M) && (4.3)\\
&\quad = \eta \sum_j P(x_n^i, \Theta_n, Y_n^k|x_{n-1}^j, M) P(x_{n-1}^j|\Theta_{1:n-1}, Y_{1:n-1}^k, M) && (4.4)
\end{aligned}
$$

In the first equation, I apply the Bayes rule to express the joint pdf of the current position and the current



(a) Position update Y^k (b) Facing direction Θ

Figure 4.2: Update models for the BOF

variables of interest Θ_n and Y_n^k at step n. The denominator $\eta = P(\Theta_n, Y_n^k|\Theta_{1:n-1}, Y_{1:n-1}^k, M)$ is just the normalizing constant that can be obtained by marginalizing the joint distribution $P(x_n^i, \Theta_n, Y_n^k|\Theta_{1:n-1}, Y_{1:n-1}^k, M)$ over all cells. η ensures that the total probability mass remains as unity.

$$\frac{1}{\eta} = P(\Theta_n, Y_n^k|\Theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.5}$$

$$= \sum_i \sum_j P(x_n^i, \Theta_n, Y_n^k|x_{n-1}^j, M) P(x_{n-1}^j|\Theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.6}$$

$P(x_{n-1}^j|\Theta_{1:n-1}, Y_{1:n-1}^k, M)$ is the probability density of previous position estimate $\overline{X}_{n-1}$. When the user takes step Y_n^k, I update the probabilities of each of the grid cells as per the user's facing direction Θ_n. The factor $P(x_n^i, \Theta_n, Y_n^k|x_n^j, M)$ incorporates the user's yaw Θ_n and step length corresponding to Y_n^k. This term propagates the probability density, akin to a transition function in a particle filter. This transition function can be factored as follows:

$$P(x_n^i, \Theta_n, Y_n^k|x_{n-1}^j, M)$$
$$= P(x_n^i, Y_n^k|x_{n-1}^j, M) P(\Theta_n|x_n^i, Y_n^k, x_{n-1}^j, M) \tag{4.7}$$

As seen in equation 4.7, the transition function can be factored into two terms. $P(x_n^i, Y_n^k|x_{n-1}^j, M)$ models the probability that cell c^i can be reached from cell c^j if the stride taken was Y_n^k, as shown in figure 4.2a. The heading transition function $P(\Theta_n|x_n^i, x_{n-1}^j, Y_n^k, M)$ models whether facing direction Θ_n is valid for the given choice of cells c^i, c^j and Y^k, 4.2b.

The step-length-based transition probability $P(x_n^i, Y_n^k|x_{n-1}^j, M)$ models the map constraints, and is non-zero only for cells c^i that can be reached from c^j. It's normalized to unity for every cell c^j.

$$\sum_i P(x_n^i, Y_n^k|x_{n-1}^j, M) = 1 \tag{4.8}$$

This ensures that every cell c^j propagates all of its probability mass to its reachable neighboring cells after step n is taken. If an angular estimate Θ_n is uniform or absent, the normalization of $P(x_n^i, Y_n^k|x_{n-1}^j, M)$ ensures that our belief that x^j was the correct probability mass at time $n-1$ doesn't change. Since all of its probability mass remains in the density after transition at step n, although distributed to its neighbors, the belief that c^j did not actually contain the user isn't being filtered out. The probability of a cell from where a step n was taken actually contained the correct belief at time $n-1$ should not change regardless of the

number of reachable neighbors it has. Otherwise, there will naturally be a bias towards cells that have more neighbors.

The filtering occurs in earnest because of the heading transition function $P(\Theta_n|x_n^i, x_{n-1}^j, Y_n^k, M)$. This function acts as a filter, selecting cells c^i that can be reached from c^j by taking a step along the facing direction Θ_n. This function behaves as an indicator and is non-zero only if the previous constraint is met. This function has the effect of maximally propagating the probabilities of cells that have the largest number of neighbors reachable by taking step Y_n^k along Θ_n. If none of its neighbors meet this criterion, none of the probability mass contained in cell c^j propagates during step n. This has the net effect of filtering out the probability that c^j truly contained the user at step $n-1$ since the user could not have taken step n from this position.

In practice, placing a finite density in the same cell even after a step is taken is useful to ensure robustness against false step detections. I also model $P(\Theta_n|x_n^i, x_{n-1}^j, Y_n^k, M)$ to account for a broad bias in estimates of the facing direction. Probability is transitioned to cells that are up to 45 degrees in angular distance away from Θ_n, forming a cone. Furthermore, the step length Y_n^k is not just a singular value. Instead, $P(x_n^i, Y_n^k|x_{n-1}^j, M)$ assigns a probability to cells that are widely distributed about a mean step distance Y_n^k.

Since the probability space is a set of grid cells over a discontinuous 2D grid, the mean position in terms of the 2D spatial coordinates of each grid cell may not lie on the grid itself. For example, if the position distribution is bimodal, with a wall between the two modes, the mean position could lie on the wall itself, which is not a valid user position. Instead, the BOF outputs the cell with the maximum probability as the current user position at step x_n.

$$x_n = \underset{i}{\operatorname{argmax}}\, P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M) \tag{4.9}$$

4.2 BOF vs PF

The advantage of using the occupancy grid filter over the particle filter is that a single transition function with support in the position space itself models the uncertainty in step detection, length, and yaw estimates. Unlike the particle filter, BOF does not suffer from the curse of dimensionality. The particle filter would require sampling particles with different step lengths and headings, which requires a large number of particles. The BOF accounts for these uncertainties by having wider transition functions $P(x_n^i, Y_n^k|x_{n-1}^j, M)$ and $P(\Theta_n|x_n^i, x_{n-1}^j, Y_n^k, M)$. The computational complexity remains the same regardless of how concentrated these transition functions are.

4.3 Fusion of other sensing modalities

The Particle Filter has historically been used as a framework for fusing multiple sensing modalities. When a sensor measurement is performed, particles can be weighted through importance sampling. By weighting particles based on how much they conform to the probabilistic model of the sensor, particles can be resampled to maximize the posterior distribution.

A similar procedure can be performed using the Bayesian Occupancy Grid Filter. If a new sensor measurement z^t is obtained at time t, it can be used to filter out grid cells that are unlikely to produce such a measurement. For example, if a high RSSI value is obtained from a beacon from a known location, the

weight of cells far away from the beacon can be decreased.

Lets assume at time t, n steps have been detected, and the current position estimate $\overline{X}_n$ conforms to the distribution $P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M)$. The likelihood that the measurement z^t was produced when the user was at cell c^i after n steps can be denoted as $P(z^t|\overline{X}_n = c^i)$ or concisely $P(z^t|x_n^i)$. Thus, the posterior distribution is now:

$$P(x_n^i|z^t, \Theta_{1:n}, Y_{1:n}^k, M) = \frac{P(z^t|x_n^i) \times P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M)}{P(z^t)} \tag{4.10}$$

4.4 Bayesian Occupancy Grid Filter without Initial Heading

In the case of the BOF and the PF for dead reckoning, the algorithms assume that both an initial estimate in the facing direction and the user's position are provided to them. The requirement for the initial position estimate can be eased as long as a user walks within the indoor environment for a long enough time for their position estimate to converge. Illustrative examples are provided in Appendix A. The sharper the user's estimate of their initial position, the faster the convergence. If the user knows their room at the initialization step, the algorithm can converge within a few steps of walking outside the room. Conversely, if the initial location provided is the entire floorplan, the user's trajectory only converges for certain indoor topologies, and the user follows a long enough trajectory.

The initial facing direction is a more rigid requirement for the initialization procedure. Without the initial heading Θ_0, the algorithm has no knowledge of the true facing direction Θ_n at a step n since its estimate is derived by adding Θ_0 to the integral of the angular velocity measurements over time. Fortunately, the facing direction can also be modeled as a random variable whose probability density is propagated over time. As a user walks within the indoor environment, mapped obstacles can filter this distribution until the facing direction estimates converge to the true facing direction.

Instead of initializing the facing direction Θ_0 along the user's true initial direction, a random variable Φ should be defined uniformly along all possible initial facing directions. Due to obvious constraints, Φ can only be initialized along a finite set of angles between $-\pi$ and π. To account for the continuous space $[-\pi, \pi)$ along which the initial angle could lie, the robustness of the BOF can be exploited. Since the BOF is robust to facing direction bias while tracking position, Θ_0 can only be assigned along a few equally spaced angles such that the angular spacing $\delta\phi$ is greater than the bias threshold. As long as the BOF can account for bias greater $\delta\phi$, $\Phi_l, l \in \{1, L\}$ only needs to be assigned along $L = \frac{2\pi}{\delta\phi}$ angles.

In summary, the joint distribution of both the current position as well as the initial heading as estimated. The distribution $P(x_n^i, \Phi_l|\theta_{1:n}, Y_{1:n}^k, M)$ is tracked which jointly characterizes joint probability of the current position and the initial facing direction. Previously, Θ_n represented the current estimate of facing direction. We now track the change in heading wrt. the initial heading at step n as θ_n. The current facing direction assuming the l^{th} initial heading is thus

$$\Theta_{n,l} = \Phi_l + \theta_n. \tag{4.11}$$

The motion model can now be modified to the following:

$$P(x_n^i, \Phi_l | \theta_{1:n}, Y_{1:n}^k, M) = \frac{P(x_n^i, \Phi_l, \theta_n, Y_n^k | \theta_{1:n-1}, Y_{1:n-1}^k, M)}{P(\theta_n, Y_n^k | \theta_{1:n-1}, Y_{1:n-1}^k, M)} \tag{4.12}$$

$$= \eta' P(x_n^i, \Phi_l, \theta_n, Y_n^k | \theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.13}$$

The distribution $P(x_n^i, \Phi_l | \theta_{1:n}, Y_{1:n}^k, M)$ in equation 4.12 is just the product distributions of the position x_n^i and initial facing directions Φ_l. This is equivalent to having L individual BOFs that separately track each with a unique initial facing direction Φ_l. The only difference lies in the normalizing constant η', which is now marginalized across all combinations of positions and initial headings.

$$\frac{1}{\eta'} = P(\theta_n, Y_n^k | \theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.14}$$

$$= \sum_l \sum_i \sum_j P(x_n^i, \Phi_l, \theta_n, Y_n^k | \theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.15}$$

The equations for motion can now be derived similarly to the regular BOF with the initial heading:

$$P(x_n^i, \Phi_l | \theta_{1:n}, Y_{1:n}^k, M)$$

$$= \eta' P(x_n^i, \Phi_l, \theta_n, Y_n^k | \theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.16}$$

$$= \eta' \sum_j P(x_n^i, \Phi_l, \theta_n, Y_n^k | x_n^j, \theta_{1:n-1}, Y_{1:n-1}^k, M) P(x_{n-1}^j, \Phi_l | \theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.17}$$

$$= \eta' \sum_j P(x_n^i, \Phi_l, \theta_n, Y_n^k | x_{n-1}^j, M) P(x_{n-1}^j, \Phi_l | \theta_{1:n-1}, Y_{1:n-1}^k, M) \tag{4.18}$$

Each BOF tracks the position x_n^i at step n independently assuming the its own unique initialization angle Φ_l. Each BOF is initialized based on the initial facing direction distribution, which can even be uniform in practice. As the user walks around the indoor environment, BOFs tracking the incorrect initial facing directions tend to collide with obstacles more than the true initial facing direction. The total cells having non-zero support decreases for the former as compared to the latter. Since the marginalization step is performed across all BOFs jointly, the total probability mass corresponding to incorrect initial facing directions decreases. As the number of steps increases, the probability mass concentrates at the true initial facing direction.

The update step is also similarly modified as:

$$P(x_n^i, \Phi_l, \theta_n, Y_n^k | x_{n-1}^j, M)$$

$$= P(x_n^i, \Phi_l, Y_n^k | x_{n-1}^j, M) P(\theta_n | x_n^i, \Phi_l, Y_n^k, x_{n-1}^j, M) \tag{4.19}$$

This approach to removing the requirement for initial position comes with the tradeoff of multiplying the computational complexity of the original BOF by the number of possible initial headings L. To make this algorithm computationally feasible for real-time applications, a fast implementation of the BOF is required.

4.5 Fast Implementation using Sparse Matrix multiplication

The Bayesian Occupancy Grid Filter tracks the probability density $P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M)$ on a grid space. The probabilities of every cell c^i can be stored in a dense vector $\hat{x}_n$ for every step n. The step-length transition function $P(x_n^i, Y_n^k|x_{n-1}^j, M)$ would then be a pre-computed matrix multiplication applied on the previous step's probability vector. Since the map is known in advance, the transition function will be a set of scalars. By substituting equation 4.7 in 4.4:

$$
\begin{aligned}
&P(x_n^i|\Theta_{1:n}, Y_{1:n}^k, M) \\
&\quad = \eta \sum_j P(x_n^i, \Theta_n, Y_n^k|x_{n-1}^j, M) P(x_{n-1}^j|\Theta_{1:n-1}, Y_{1:n-1}^k, M) && (4.20) \\
&\quad = \eta \sum_j P(x_n^i, Y_n^k|x_{n-1}^j, M) P(\Theta_n|x_n^i, Y_n^k, x_{n-1}^j, M) P(x_{n-1}^j|\Theta_{1:n-1}, Y_{1:n-1}^k, M) && (4.21)
\end{aligned}
$$

Which in vector form can be written as:

$$
\begin{aligned}
\hat{x}_n^i &= \eta \sum_j P(x_n^i, Y_n^k|x_{n-1}^j, M) P(\Theta_n|x_n^i, Y_n^k, x_{n-1}^j, M) \hat{x}_{n-1}^j && (4.22) \\
&= \eta \sum_j y_{i,j}^k P(\Theta_n|x_n^i, Y_n^k, x_{n-1}^j, M) \hat{x}_{n-1}^j && (4.23)
\end{aligned}
$$

The normalizing constant can simply be applied by dividing $\hat{x}_n$ by the sum of its entries $\sum_i \hat{x}_n^i$.

The only issue holding us back from implementing the BOF as a matrix multiplication is the term $P(\Theta_n|x_n^i, Y_n^k, x_{n-1}^j, M)$ which is unique to every facing direction $\Theta_n \in [-\pi, \pi)$. This function would have to be computed for every pair of i, j during every transition.

To alleviate this issue, the robustness of the BOF can be exploited again, similar to how BOF was performed without initial heading estimates. Instead of defining a heading transition function for every angle, only a fixed number of transition functions need to be defined for a set of discrete angles. The transition function of the heading closest to the current facing direction can then be used.

Mathematically, this can be described using the BOF with no initial heading. Separate probability vectors can be defined for every initial angle $\hat{x}_n^l$. Using the transition function in equation 4.19, and substituting it into 4.18,

$$
\hat{x}_n^{l,i} = \eta' \sum_j y_{i,j,l}^k P(\theta_n, \Phi_l|x_n^i, Y_n^k, x_{n-1}^j, M) \hat{x}_{n-1}^{l,j} \tag{4.24}
$$

The step transition function $y_{i,j,l}^k$ is equal to $y_{i,j}^k$ since it doesn't consider the heading angle as described in section 4.1. The heading transition function $P(\theta_n, \Phi_l|x_n^i, Y_n^k, x_{n-1}^j, M)$ can be written as $P(\Theta_{n,l}|x_n^i, Y_n^k, x_{n-1}^j, M)$, where $\Theta_{n,l}$ follows equation 4.11.

The set of angles for which there is a predefined transition function are defined as $\Psi_s, s \in \{1, S\}$. Instead of directly using $P(\Theta_{n,l}|x_n^i, Y_n^k, x_{n-1}^j, M)$, $P(\Psi_s|x_n^i, Y_n^k, x_{n-1}^j, M)$ can be used, where

$$\Psi_s = \min_{\Psi_r} |\Theta_{n,l} - \Psi_r|, \ r \in \{1, S\} \tag{4.25}$$

Thus, the precomputed transition function can be defined as

$$\hat{x}_n^{l,i} = \eta' \sum_j y_{i,j}^k \alpha_{i,j}^s \hat{x}_{n-1}^{l,j} \tag{4.26}$$

with a unique transition matrix $F^{s,k}$ with elements $f_{i,j}^{s,k}$ for every step length Y^k and heading angle Ψ_s

$$f_{i,j}^{s,k} = y_{i,j}^k \alpha_{i,j}^s \tag{4.27}$$

$$\hat{x}_n^l = F^{s,k} \hat{x}_{n-1}^l \tag{4.28}$$

When the initial heading is present, only one probability vector $\hat{x}_n^l$ needs to be transitioned along the angle Ψ_s closest to the current heading Θ_n.

4.5.1 Exploiting sparsity

The total number of cells needed to span an entire indoor environment is usually of the order $O((\frac{L}{\delta l}^2))$, where L is the length of the indoor environment, and δl is the length of a cell. This number can be around 16000 for a reasonably sized environment. The transition function involves multiplying a vector of this length by a square matrix, which would be far too big for real-time applications.

But as shown in figure 4.2a, the number of cells a user can reach after taking a step is far smaller. The step transition function $y_{i,j}^k$ is non-zero only for a limited number of neighboring cells. This the matrices $F^{s,k}$ are extremely sparse in practice, with not more than 25 elements per row. The BOF can now be expressed as a matrix multiplication between a sparse matrix and a dense vector. Sparse matrix multiplications can be performed using highly optimized libraries for even large matrices.

Chapter 5

Evaluation

To show the effectiveness of the proposed Bayesian Occupancy Grid Filter, its robustness is evaluated for errors in step detection, length, and facing direction. Given an accurate indoor floor plan and a step model of the user, the BOF algorithm can handle errors far better than a PF. The experiments were conducted on the second floor of the Coordinated Science Lab at UIUC. Its floor plan was used to determine the free areas where users could navigate by discretizing the space. Barriers were modeled wherever walls and other obstacles were present in the floor plan. These barriers served to filter the PF whenever particles crossed them. In the case of the BOF, the discretization was done using a grid, with each cell being $\delta l \times \delta l$, equal to 30cm by 30cm. The grid has a size of 212 by 82 cells, which gives 17383 cells in total. Cells that contained barriers were constrained to have 0 probability.

In our comparisons with the PF, the number of particles used was kept the same as the total number of cells to ensure parity in computational complexity. Each cell stores a double value to indicate the probability of the user being in the cell.

5.1 Simulation

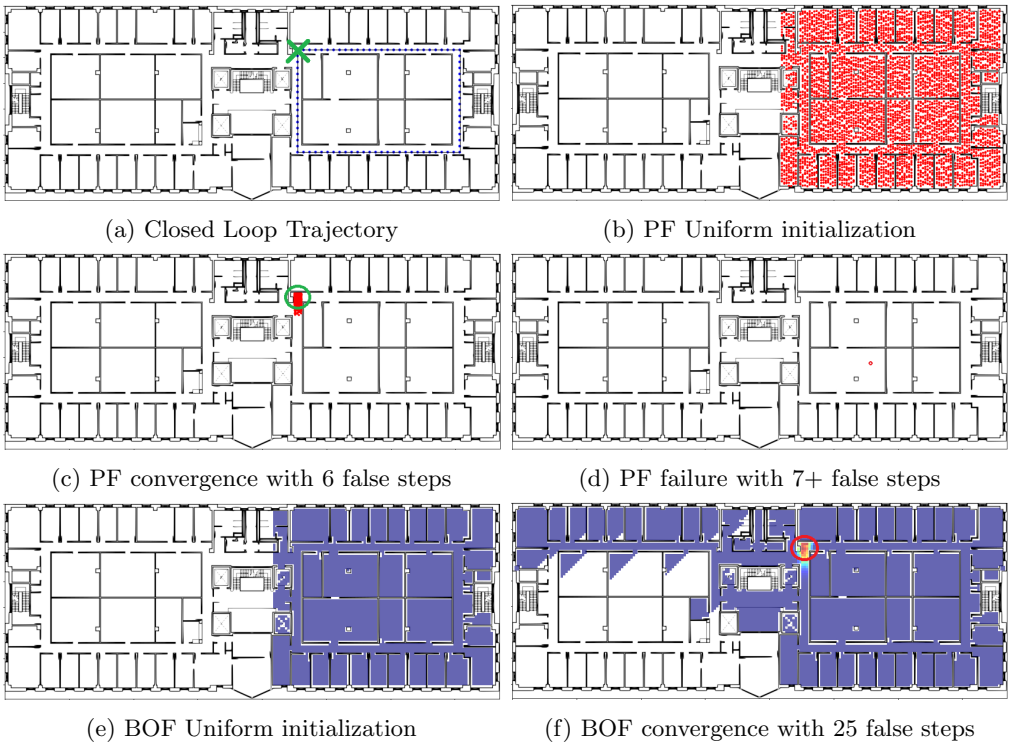
Experiments were conducted by simulating fixed trajectories along the floorplan. Each trajectory used for the experiments had a fixed step length, step count, and facing direction to simulate ideal ground-truth conditions. They were also initialized from fixed starting points distributed across the floorplan.

During the simulations, each trajectory was initialized in a uniform area that included the true starting point, and these areas ranged from the entire floor to a small room. This reflects real-world scenarios where a user may not have access to the algorithm's coordinate system and exact starting location measurements. These trajectories and their initialization areas are described in Appendix A.

First, an example is shown of how both the PF and the BOF are robust to false step detection in simulation. In figure 5.1, this evaluation process is illustrated. The reference is first shown closed-loop trajectory used, consisting of 88 steps. The reference closed-loop trajectory, marked with blue, shown in 5.1a, starts and ends at the top left corner of the path, marked by the green cross. I initialize the user's location uniformly in a large indoor area for both algorithms.

When either algorithm is provided accurate step detects, step lengths, and facing directions, they converge to the correct final location. If the number of false step detections increases, the PF is found to be only resilient to six false step detections before it diverges. On the other hand, the BOF can handle around 25

false step detections. In the example, the number of particles used for the PF is 17000, and all grid cells are active in the case of the BOF.



(a) Closed Loop Trajectory (b) PF Uniform initialization

(c) PF convergence with 6 false steps (d) PF failure with 7+ false steps

(e) BOF Uniform initialization (f) BOF convergence with 25 false steps

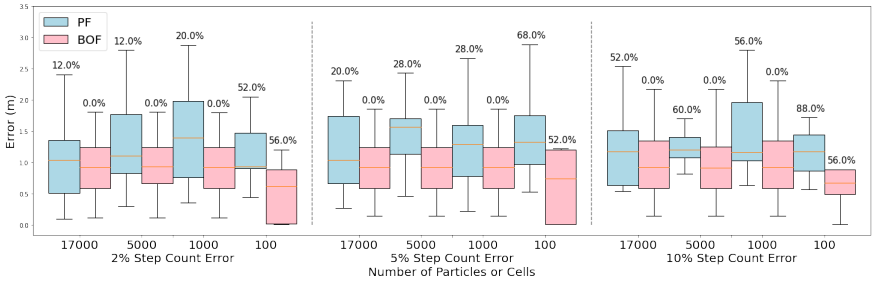
Figure 5.1: Illustration of the robustness of PF and BOF against false step detections. In the case of the PF, each red dot represents a particle, while for the BOF, the colored regions represent cells with finite support.

5.1.1 Robustness

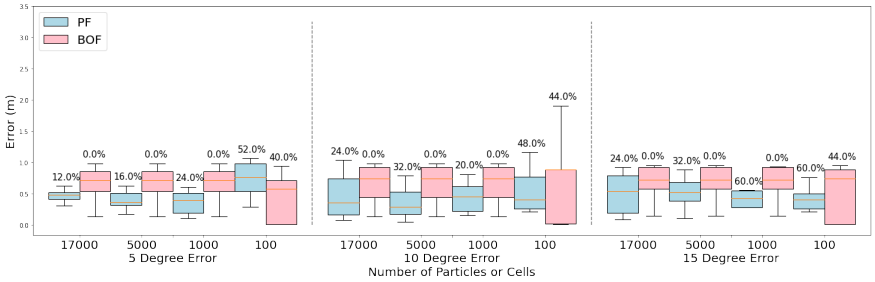
Figure 5.2 shows a formal comparison of the two algorithms. The performance comparison uses a predefined set of five trajectories. In the trajectory shown in 5.1, the initialization is half the floor plan, and the route is closed-loop. The algorithms are evaluated by adding a random number of false step detects and random biases to independently simulate each type of inaccuracy that appears in the real world. The other four trajectories are described in Appendix A.

During testing, the objective of the algorithms was to converge to the true endpoint at the end of the trajectory, and the reported result measures the Euclidean error between the true endpoint and the location at which the algorithm converged. Experimental samples where either algorithm converges to a point 3m away from the endpoint were marked as diverged. In the case of the Particle Filter, when divergence occurs, all particles tend to be eliminated, requiring resampling. When this occurs, the previous sample mean of the particles is reported as the final location. This is then used to compute the error.

Both algorithms are compared based on the number of particles used for the PF and the active cells for the BOF. For the BOF, the active cell count was maintained by pruning cells below a threshold determined by ordering the cells based on their probability. A larger number of particles/active cells provides a larger support for the position to ensure robustness at the cost of computational requirements.



(a) Robustness to False Step Detection



(b) Robustness to Facing Direction Bias

Figure 5.2: Comparison of the robustness of PF and BOF against false step detections and bias in facing direction.
Only samples that have converged have been included in the boxplots, and
the percentages above the boxplot whiskers represent the fraction of samples that diverged

5.1.2 False Step Detection and Length

First, the algorithms' robustness to false step detection is compared in figure 5.2a. Both algorithms were tested by adding or subtracting a random number of steps from the true trajectories while maintaining the correct facing direction and step length. All five trajectories were simulated five times, resulting in 25 runs for every combination of particle/cell count and error percentage. The boxplots for all these runs are plotted after pruning the cases where the filters diverged. The percentage of cases where the filters diverged is mentioned above the whiskers.

These results also capture the algorithms' robustness against step length variance. Since we use a finite number of fixed templates for the various step types, the updates are performed with specific step lengths for both filters. The variance in the user's true step length arises in the form of either more or fewer step detections due to the user traveling a variable length distance with each step. If the user travels a greater distance with each step than the length according to the original step model, the total number of steps needed to traverse a corridor will be less than expected, and vice-versa. Thus, robustness to false step count enables robustness against step length variance.

As the total number of particles decreases, the error increases for the PF. Furthermore, the PF consistently diverges in all scenarios. We often observe this happening in cases where the initialization area was large and false steps were detected early. In contrast, the error in the case of the BOF remained consistently lower than the PF. In the case of 100 active cells, we observe a lower error in the converged samples because of the smaller active support, but this reduces robustness, resulting in some samples diverging. However, we see no divergence up to 1000 active cells in the case of the BOF.

Figure 5.3: BOF convergence with no initialization for Trajectory 1

5.1.3 Direction bias

The algorithms' robustness to bias in facing direction estimates is compared in figure 5.2b. I tested both algorithms by adding a consistent facing direction bias to one path of each trajectory while maintaining true step count and length. All five trajectories were simulated four times, resulting in 20 runs for every combination of particle/cell count and error percentage.

Unlike in the case of false step count, we see the error remains consistent across all particle counts and active cells in the converged samples. The PF consistently has a lower error than the BOF after convergence. But unlike the BOF, we see some samples diverging regardless of particle count for the PF. This divergence worsens as the number of particles decreases. We only observe divergence when the number of active cells for the BOF is 100.

5.1.4 Convergence with no facing direction initialization

Next, the case where the initial heading is no longer provided is simulated. I show how the BOF converges when no initial heading information is available. In the previous section, the BOF was shown to converge after the user walks around an indoor environment, even if their original position was initialized uniformly throughout the indoor floor plan. This convergence occurs due to the asymmetry in the floorplan that is exploited by choosing a specific trajectory. Although the user's location is uniformly initialized, only one possible initial location could permit such a trajectory in the given environment. The BOF is able to converge onto the correct final location corresponding to the trajectory taken after starting at the only possible initial location.

The same principle applies to the case where no initial heading is provided. Unless a trajectory and facing direction pair are unique to an initialization, the BOF cannot converge after the trajectory. If multiple initial position/facing direction pairs exist, the BOF converges to a multimodal distribution corresponding to all of the valid positions. To illustrate this phenomenon, I run the BOF for the scenarios in A.1 and A.4.

In the first case, the convergence of the BOF is shown in figure 5.3. Here, we see a bimodal distribution. This is because the trajectory, shown in figure A.1a, is a closed loop. The starting position could have been at two positions. If the starting/ending position were at the green cross at the top left of the path, the initial facing direction would have been towards the right. If the starting position were at the bottom right, the initial facing direction would have been towards the left. Both are valid for the given trajectory. Since the position was initialized uniformly for both starting positions, the BOF converges to both permissible locations instead of converging to a single position.

We see convergence in the second scenario shown in figure 5.4, corresponding to A.4. Given the uniform

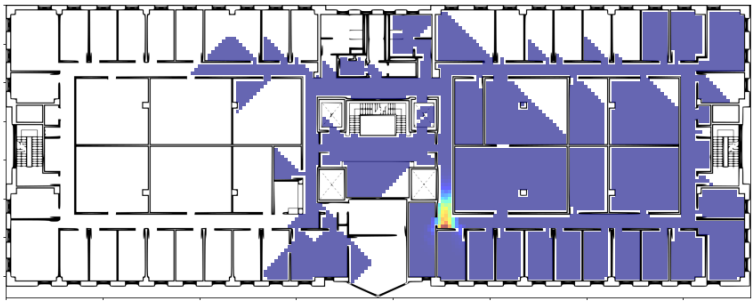
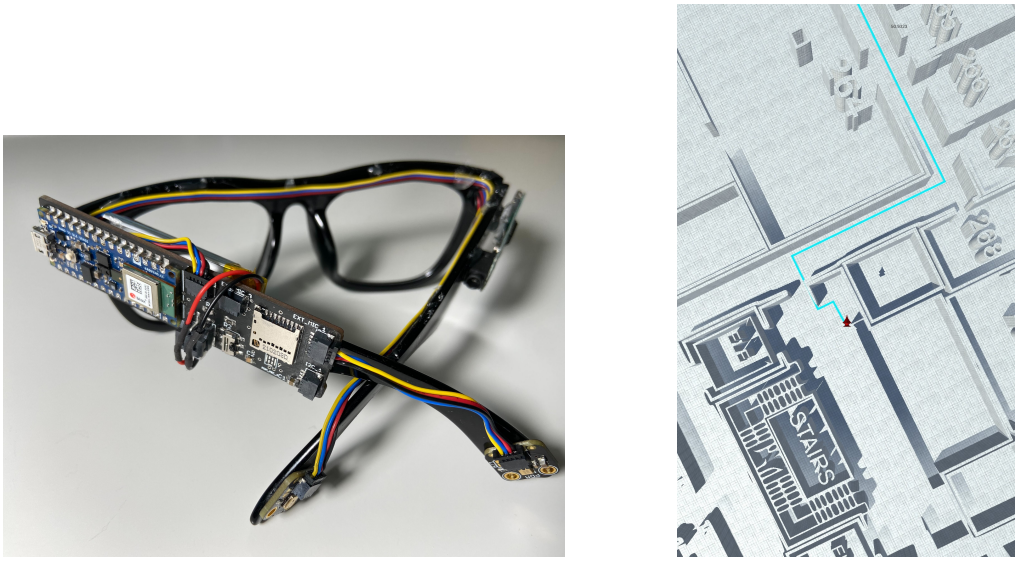


Figure 5.4: BOF convergence with no initialization for Trajectory 4



(a) Smartglass platform (b) Android application

Figure 5.5: Real-time System

initialization, this is because there is only one valid combination of initial position and heading from which the trajectory could have been traversed. The BOF correctly converges to the final position, denoted by a red cross, shown in figure A.4.

5.2 Real-time System Implementation

The system was developed on the Android platform to run the core algorithms and the user interface. The core algorithms were implemented on the native C++ layer. The UI was implemented on the Unity 3D renderer. The sensors consisted of the inbuilt IMU of the Android Smartphone and a pair of smart glasses with an integrated IMU. The glass was equipped with a Bluetooth transmission radio, through which it could communicate with the smartphone. Finally, an HRTF audio-cue navigation system was developed to serve as an additional modality for user interaction. These audio cues simulate directional sound, as though a virtual assistant guides the user.

Chapter 6

Conclusion

In this thesis, we have seen that indoor localization for a pedestrian is possible by performing dead reckoning with just the built-in IMU of a personal device. Given the layout of the indoor environment in advance, it is possible to determine the user's position without any external infrastructure. The two algorithms described, namely the Particle Filter and the Bayesian Occupancy Grid Filter, can both perform reasonably well in low error regimes.

Even without any initial estimates of the user's position, both algorithms can converge to the user's true position after they traverse a reasonable distance within the environment. But when we operate in high error regimes, i.e., when there is an error in estimating step count, step length, and facing direction, the BOF can still perform exceptionally well, unlike the PF. The particle filter is especially prone to diverge because of fall step detections, as it may no longer place support at the user's true location.

To demonstrate its effectiveness, the BOF is shown to ensure robustness for indoor localization in scenarios with low measurement accuracy. Compared to the PF, which has been previously used extensively for indoor localization, the BOF is far more robust to false step detections, step length inaccuracies, and facing direction estimates. In challenging scenarios where the user's position is uniformly initialized throughout the entire indoor map, the BOF doesn't diverge even when the number of active cells is just 1000. This is in contrast to the PF, which diverges even when there are 17000 active particles, even in realistic scenarios where a user is initialized within a room-sized area.

One of the most common assumptions for navigation is that the initial facing direction is provided to the algorithm. In outdoor navigation systems, magnetometers are commonly used to estimate this by tracking the Earth's magnetic field. However, magnetometers are not reliable for indoor navigation systems. Magnetic anomalies contaminate the Earth's magnetic field, preventing its estimation. Instead, indoor navigation systems rely on some other external infrastructure or user input to initialize this variable.

The BOF provides a computational framework that allows us to challenge this assumption. The initial heading can be tracked as a random variable, and the map information can be used to converge to its true value if the trajectory the user takes doesn't have other symmetric candidates.

Although this brute-force approach may not seem possible in real-time applications, this is shown not to be true. The facing direction is a random variable that can take any value between $-\pi$ and π. However, because of the robust BOF, we only need to track a small number of heading initialization values within this interval. By optimally tracking equally spaced heading directions, we can still achieve convergence.

Furthermore, the BOF lends itself to be framed as a sparse matrix-multiplication problem. Unlike a PF,

where a resampling step must be performed after every step transition, the BOF only requires the probability vector representing the cell probabilities to be multiplied by an appropriate sparse matrix. Sparse matrix multiplication is a well-studied problem in computational hardware, and most architectures offer optimized libraries to perform it.

The appeal of the PF is the capability to fuse other sensing modalities quite easily. This is shown to be true even of the BOF. The procedure is similar to the PF, where the posterior probability of the user present at a certain cell is derived given a sensor measurement. We also have the benefit of skipping any resampling requirement for the BOF.

Thus, we can conclude that the BOF is better suited to be the framework of choice for Pedestrian Indoor Localization than the PF. In the high error regimes of Pedestrian Dead Reckoning, where only an IMU is the sensing modality, the BOF manages to stay robust to sensing errors. Furthermore, its low complexity in implementation requirements and computation makes it well-suited for real-time requirements.

The real-time performance of the BOF is also demonstrated on an Android-based prototype. This application is able to navigate a user to their destination within an indoor layout of the Coordinated Science Lab at UIUC using the IMU built into a wearable glass prototype.

References

[1] X. Li, D. Zhang, Q. Lv, *et al.*, "Indotrack: Device-free indoor human tracking with commodity wi-fi," vol. 1, no. 3, 2017. DOI: 10.1145/3130940. [Online]. Available: https://doi.org/10.1145/3130940.

[2] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level localization with a single wifi access point," ser. NSDI'16, Santa Clara, CA: USENIX Association, 2016, pp. 165–178, ISBN: 9781931971294.

[3] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and i can track my user's arm," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '16, Singapore, Singapore: Association for Computing Machinery, 2016, pp. 85–96, ISBN: 9781450342698. DOI: 10.1145/2906388.2906407. [Online]. Available: https://doi.org/10.1145/2906388.2906407.

[4] S. Beauregard, "A helmet-mounted pedestrian dead reckoning system," in *3rd International Forum on Applied Wearable Computing 2006*, 2006, pp. 1–11.

[5] Z.-A. Deng, G. Wang, Y. Hu, and D. Wu, "Heading estimation for indoor pedestrian navigation using a smartphone in the pocket," *Sensors*, vol. 15, no. 9, pp. 21 518–21 536, 2015, ISSN: 1424-8220. DOI: 10.3390/s150921518. [Online]. Available: https://www.mdpi.com/1424-8220/15/9/21518.

[6] H. Xie, T. Gu, X. Tao, H. Ye, and J. Lv, "Maloc: A practical magnetic fingerprinting approach to indoor localization using smartphones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14, Seattle, Washington: Association for Computing Machinery, 2014, pp. 243–253, ISBN: 9781450329682. DOI: 10.1145/2632048.2632057. [Online]. Available: https://doi.org/10.1145/2632048.2632057.

[7] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12, Istanbul, Turkey: Association for Computing Machinery, 2012, pp. 293–304, ISBN: 9781450311595. DOI: 10.1145/2348543.2348580. [Online]. Available: https://doi.org/10.1145/2348543.2348580.

[8] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," ser. MobiSys '12, Low Wood Bay, Lake District, UK: Association for Computing Machinery, 2012, pp. 197–210, ISBN: 9781450313018. DOI: 10.1145/2307636.2307655. [Online]. Available: https://doi.org/10.1145/2307636.2307655.

[9] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Complementary filter design on the special orthogonal group so(3)," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 1477–1484. DOI: 10.1109/CDC.2005.1582367.

[10] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance," in *2007 IEEE 23rd International Conference on Data Engineering*, 2007, pp. 1046–1055. DOI: 10.1109/ICDE.2007.368963.

[11] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. [Online]. Available: https://doi.org/10.1115/1.3662552.

[12] D. He, W. Xu, and F. Zhang, *Kalman filters on differentiable manifolds*, 2021. arXiv: 2102.03804 [cs.RO].

[13] M. F. Mansour and D. W. Waters, "Map-assisted kalman filtering," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 3208–3212. DOI: 10.1109/ICASSP.2013.6638250.

[14] J. S. Liu and R. Chen, "Sequential monte carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998, ISSN: 01621459. [Online]. Available: http://www.jstor.org/stable/2669847 (visited on 11/25/2023).

[15] F. van Diggelen and P. K. Enge, "The world's first gps mooc and worldwide laboratory using smartphones," 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:63645012.

Appendix A

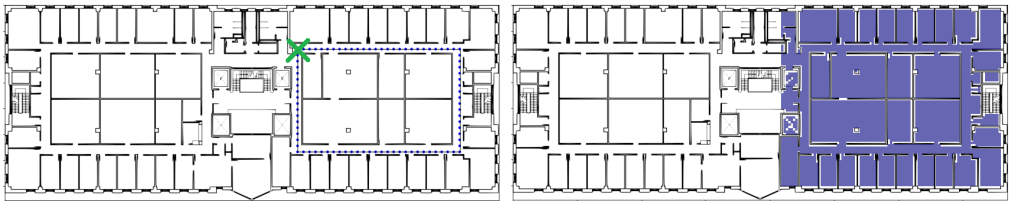
Simulation Trajectories

Here are the five trajectories used to perform the simulation experiments for evaluating the BOF for Robustness against false step detections and facing direction bias. In all cases, the green cross denotes the true starting position, while the red cross denotes the end position. Each blue dot along the trajectory represents a step being taken.

Both the BOF and the PF are initialized in the area denoted covered with blue. All the grid cells within the blue area are initialized with uniform probability for the BOF. On the other hand, particles are randomly sampled within the blue area in the case of the PF.

A.1 Trajectory 1

This trajectory represents a closed loop, so both the initial and the final location in the trajectory are the same. The user is initialized to half the area of the indoor map, presenting a challenging scenario for both algorithms.

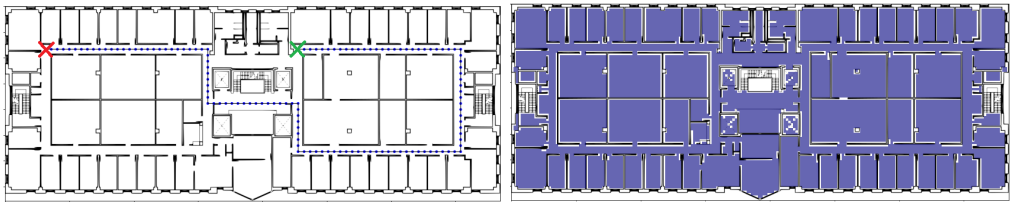


(a) Trajectory (b) Uniform initialization area

Figure A.1: Trajectory 1

A.2 Trajectory 2

This trajectory is the most challenging scenario tested, where the user is initialized uniformly within the entire indoor map. The effectiveness of the BOF is showcased in this scenario, where the filter converges to the user's location after traversing along a long trajectory, even with the presence of uncertainties.

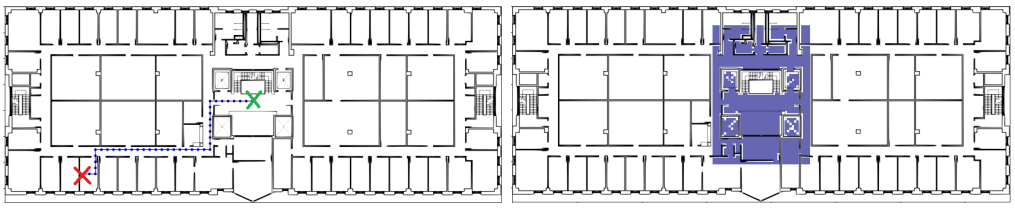


(a) Trajectory (b) Uniform initialization area

Figure A.2: Trajectory 2

A.3 Trajectory 3

This trajectory is a simpler scenario, with the initialization area being about a fourth of the total indoor map. But the endpoint is in a room, an area that has a replicated type of topology. False steps cause the position estimates to diverge to a different room incorrectly.

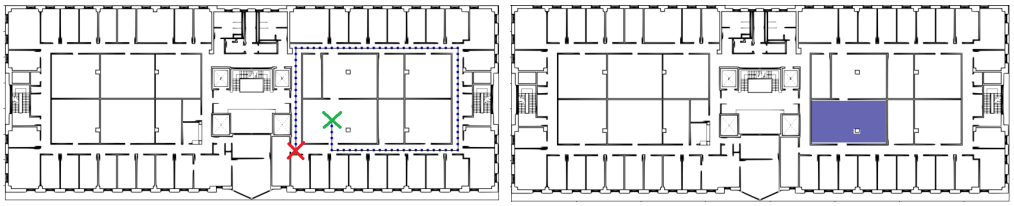


(a) Trajectory (b) Uniform initialization area

Figure A.3: Trajectory 3

A.4 Trajectory 4

The previous cases represented challenging scenarios where the user doesn't know their starting location. This case represents a more reasonable use case where the user knows what room they are present in and initializes their location to that room.

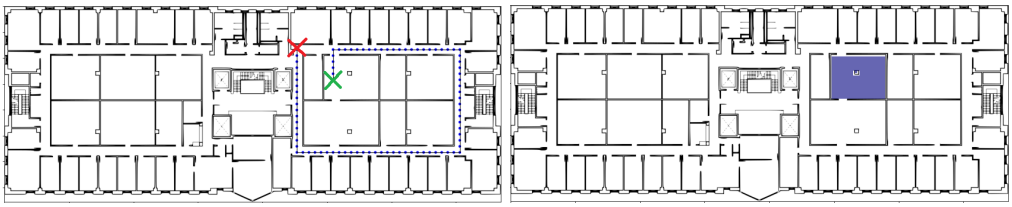


(a) Trajectory (b) Uniform initialization area

Figure A.4: Trajectory 4

A.5 Trajectory 5

This is another case where the user knows which room they are currently in. This room is smaller than the previous case and is a more forgiving scenario.



(a) Trajectory (b) Uniform initialization area

Figure A.5: Trajectory 5