TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Realzeit-Computersysteme

Crowdsourcing Motion Maps based on FootSLAM for Reliable Indoor Pedestrian Navigation in Multistory Environments

Maria Jesus Garcia Puyol

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Klaus Diepold
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Sc. Samarjit Chakraborty
	2. Prof. Dr. Jose Neira Parra, Universidad de Zaragoza, Saragossa, Spanien

Die Dissertation wurde am 08.06.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 09.12.2016 angenommen.

Acknowledgements

In September 2011, after receiving my degree in Telecommunications Engineering, Dr. Patrick Robertson encouraged me to pursue my PhD studies to expand my research on the topic of pedestrian navigation. This work led to the greatest adventure of my life: joining Google in February 2014.

I would like to express my immense gratitude to Patrick for mentoring me through both my Master and PhD theses and encouraging me to always learn new things. Patrick, your enthusiasm for science is contagious and your brilliant ideas, advice and guidance were invaluable. I am eternally grateful for your precious time.

I am deeply thankful to Prof. Samarjit Chakraborty for guiding me throughout the preparation of this thesis, and to Prof. Neira for reviewing and evaluating it.

I would like to thank DLR and Institute for Real-Time Computer Systems at the Technical University of Munich for giving me the opportunity to work on my PhD thesis and Mrs. Dippold and Mrs. Hofstetter for helping me with all my bureaucratic questions. I would also like to thank DAAD for providing me with an outstanding scholarship to pursue my PhD studies in Germany.

For the research presented in this thesis I have collaborated closely with many colleagues at DLR. I would like to express my gratitude to Oliver, Korbinian, Cristina, Susanna, Thomas, Mohammed and Luigi. I would also like to thank Christian and Waldi for making my stay at DLR a remarkable experience and Jutta Uelner and Christina Burger for helping me with all administrative matters.

My friends and family have constantly encouraged me to finish writing this thesis, especially after moving to the USA to work at Google. Thanks to my amazing relatives for always making me laugh. Thanks to Carmen, Elena, Alba and Alexia for dancing with me like there is no tomorrow. Thanks to Carlos, Ester and Mari Angeles for becoming my best friends in Munich.

My twin sister Ana is the best life partner I could ever wish for. Ana, thanks for always understanding me and giving me strength when I need it the most. Miguel and Julia, thanks for your constant encouragement and excitement. Daniel, my amazing boyfriend, thanks for your patience and constant loving support.

This thesis is dedicated to my parents, Jose Miguel and Maria Jesus. Mom, dad, you have always encouraged me to pursue my dreams and follow my passions and you made it possible for me to start and finish this PhD dissertation. I will be forever thankful.

Zusammenfassung

Laut der amerikanischen Umweltschutzbehörde und der Europäischen Kommission verbringen Amerikaner und Europäer 90% ihrer Zeit innerhalb von Gebäuden. Reisende am Flughafen, Besucher im Museum oder Feuerwehrleute während einer Rettungsaktion würden von hochgenauen Ortungsmöglichkeiten innerhalb von Gebäuden profitieren. Im Gegensatz zu Ortungslösungen im Freien, stellt die Ortung in geschlossenen Räumen eine besondere Herausforderung dar, weil satellitenbasierte Navigationsmethoden keine exakte Positionierung ermöglichen und detaillierte Karten oftmals nicht verfügbar sind.

FootSLAM ist eine auf Bayes'scher Theorie basierende Methodik, die es ermöglicht eine Karte der Wahrscheinlichkeitsverteilung von Schritten eines Fußgängers in einer Umgebung anzunähern, bei gleichzeitiger Abschätzung der Position und Orientierung des Fußgängers während dessen Fortbewegung. Um dies zu realisieren benötigt FootSLAM ausschließlich Messungen der Schritte des Fußgängers - der sogenannten menschlichen Odometrie. Diese Messungen können z.B. mittels einer am Fuß getragenen inertialen Messeinheit oder der Sensorik eines Smartphones gesammelt werden.

Der FeetSLAM Algorithmus erweitert FootSLAM durch Crowdsourcing indem die Aufgabe der Kartierung an eine Gruppe aus kollaborierenden Fußgängern übergeben wird, welche aktiv oder passiv Schrittmessungen in der gleichen Umgebung sammeln. FeetSLAM kombiniert die resultierenden individuellen FootSLAM Karten, um eine umfangreichere und genauere Karte der "begehbaren"Bereiche eines Gebäudes zu generieren.

Die vorliegende Arbeit erweitert und verbessert sowohl FootSLAM als auch FeetSLAM, um die Lücke hin zu vollautomatisierter Gebäudekartografie mittels Crowdsourcing zu schließen. Das Ziel ist es Benutzern Kartenmaterial mit einer Genauigkeit von weniger als einem Meter zur Verfügung zu stellen, welches die Lokalisierung in Flughäfen, Museen oder Einkaufszentren verbessert, die Zeit bis zur Auffindung von Opfern nach einem Notruf verkürzt oder zur Koordination von Feuerwehrleuten während einer Rettungsaktion beitragen kann.

Die Hauptbestandteile dieser Dissertation lauten wie folgt: 1) Erweiterung von FootSLAM auf mehrstöckige Umgebungen, 2) Reduktion der Rechenkomplexität von FootSLAM von $\mathcal{O}(t^2)$ auf $\mathcal{O}(t \log t)$, um die Verarbeitung von größeren Bereichen in Echtzeit zu vereinfachen, 3) Reduktion der Rechenkomplexität der Kartenkombination von FeetSLAM, 4) Analyse der Voraussetzungen und Anwendungsmöglichkeiten großflächiger Kartierung.

Nach dem Wissen der Autorin ist dies die erste Arbeit welche die Möglichkeit der Kartierung von großen, mehrstöckigen Umgebungen in einer kollaborierenden Art und Weise und unter Verwendung einzig von Inertialsensorik aufzeigt.

Abstract

According to the United States Environmental Protection Agency and the European Commission, Americans and Europeans spend 90 % of their time in indoor environments. Travelers in an airport, visitors in a museum or firefighters during a rescue mission would highly benefit from high-accurate indoor positioning. However, and in contrast to outdoor navigation, indoor navigation is particularly challenging because satellite-based navigation techniques do not yield accurate positions and precise indoor plans are often unavailable.

FootSLAM is a technique rooted in Bayesian theory that estimates the pose (position and heading) of a pedestrian as he walks in a constrained environment while it simultaneously builds a map of step direction probabilities at the visited areas. To do this, FootSLAM only needs measurements of the pedestrian's steps – a.k.a. human odometry. These measurements may be collected for example by means of a foot-mounted Inertial Measurement Unit (IMU) or using the sensors of a smartphone.

The FeetSLAM algorithm crowdsources FootSLAM by assigning the mapping task to a group of collaborating pedestrians who actively or passively collect step measurements within the same environment. FeetSLAM merges the resulting individual FootSLAM maps to generate a more complete and accurate map of the "walkable" areas.

This thesis extends and improves both FootSLAM and FeetSLAM to bridge the gap towards fully automated crowdsourced indoor mapping. The goal is to provide users with sub-meter accurate maps to improve localization in airports, museums or shopping centers, reduce the time needed to find victims after an emergency call or coordinate a firefighter team during a rescue mission.

The original contributions of this thesis are: 1) extending FootSLAM to multistory environments, 2) reducing FootSLAM's computational complexity from $\mathcal{O}(t^2)$ to $\mathcal{O}(t \log t)$ to facilitate real-time processing of larger areas, 3) reducing the computational complexity of the map combination step of FeetSLAM from quadratic to linear in the area of a floor, and 4) analyzing the requirements and applications of large-scale mapping.

To the knowledge of the author, this is the first time to have shown the possibility of mapping large multistory environments in a collaborative fashion using only inertial sensors.

Contents

1 Introduction

Humans have always been interested in knowing where they are and what their surroundings have to offer. When outdoors and in good Line-of-sight (LoS) conditions, pedestrians rely on Global Navigation Satellite System (GNSS) receivers to estimate their position and navigate from one place to the next. However, indoor pedestrian navigation is a challenging problem because inside buildings and underground, satellite-based positioning systems are not accurate enough. And yet, according to the United States Environmental Protection Agency (EPA) [160] and the European Commission [47], Americans and Europeans spend 90 % of their time in indoor environments. As a result, there is a strong need for indoor location-based services in two main domains:

1. **Mass market applications**: Multimodal transportation networks and interconnection points often require pedestrians to navigate their way through extensive indoor and underground areas sometimes with outdoor transitions. For such people on the move, smartphones, tablets and similar devices have become the central point of reference in terms of communications and information services, for example in an airport, in a mall or in a museum.

2. **Emergency and situation awareness applications**: On December 3, 1999, six firefighters were killed during a fire in an abandoned six-story building in Massachusetts, USA [5]. One of the causes of the accident was that "even to long term employees, the building was hard to navigate" [5], which made the firefighters get lost trying to rescue two homeless people. In security operations or rescue missions, knowing each team member's relative position can increase situation awareness and help coordinate the team. Similarly, 135 million people who are visually impaired worldwide [159] are in need of a reliable pedestrian navigation system to undertake daily tasks.

This thesis extends and improves two techniques, **FootSLAM** and its collaborative approach **FeetSLAM**, to build maps of step direction probabilities in a real-time collaborative fashion. In turn, these maps may be used for improved map-aided indoor navigation, allowing the development of mass-market and emergency applications.

1.1 The Challenges of Indoor Pedestrian Navigation

Accurate outdoor positioning solutions rely on GNSS signals (e.g. those collected by the Global Positioning System (GPS) receiver of a smartphone). However, inside a building,

in an urban canyon or underground, the presence of walls and other obstacles causes diffraction, refraction and multipath effects, or losing LoS to a sufficient number of satellites. This usually results in satellite signal degradation and can reduce drastically the positioning accuracy [139].

1.1.1 Infrastructure-based vs. Infrastructure-free Techniques

Two main alternatives have been proposed to address the deficiencies of satellite signals for indoor pedestrian navigation (see [51] for a survey on these alternatives for emergency responders):

1) Infrastructure-based techniques, for example based on Received Signal Strength (RSS) from Wireless Local Area Network (WLAN) access points [96], Radio Frequency Identification (RFID) tags [163], Ultra Wide Band (UWB) transmitters, Bluetooth Low Energy (BLE) transmitters [49], wide area network nodes [118], cell towers [99], etc.

2) Infrastructure-free techniques such as Pedestrian Dead Reckoning (PDR) systems [71]. PDR consists of estimating the pose (3D position and heading) of a pedestrian at each step by integrating the cumulative change in pose since the last step. To estimate steps, PDR relies on inertial sensors (accelerometers, gyroscopes) – and sometimes additional sensors like cameras, laser scanners, altimeters or magnetometers. Because of their less privacy-invasive nature, accelerometers, gyroscopes and magnetometers — nowadays present in all modern smartphones or integrated in an Inertial Measurement Unit (IMU) — have been widely adopted to measure pedestrian steps. The accelerometer measures linear acceleration, the gyroscope measures turn rates and the magnetometer measures the magnetic field. Double integrating the acceleration yields position change; integrating the turn rate yields attitude change. The magnetometer is typically used to correct the biases of the gyroscope [1][166].

The main drawback of infrastructure-based techniques is that the infrastructure needs to be installed and maintained. In addition, after an earthquake or during a fire, such infrastructure might not be available. In contrast, inertial sensors and those often used in conjunction (altimeter, magnetometer) have the great advantage that they are now integrated in mobile devices (phones, tablets, watches, etc.) and thus highly ubiquitous. The main disadvantage of pedestrian navigation based on inertial sensors is that it lacks an anchor to a global coordinate system: The pose of the pedestrian can only be estimated relative to his initial pose.

FootSLAM relies solely on human odometry measurements collected by means of a sensor array containing the following: a triaxial accelerometer, a triaxial gyroscope and a triaxial magnetometer. These sensors have a relatively low data rate, making both server based positioning and on-device processing viable options. In addition, in pedestrian navigation that is targeted to giving people reliable and useful walking directions, it is often forgotten that in addition to *body level accuracy*[1)] one requires accurate orientation information in

[1)] By this we mean a targeted position estimate with an accuracy of roughly 0.5 to 1 m.

order to know which way the person is facing. Given an absolute initial heading estimate, human odometry intrinsically provides the orientation of the pedestrian by nature of its close coupling to a model of the human body and pedestrian motion.

The FootSLAM framework, however, allows fusing other sensor measurements, when available, to improve its accuracy. These measurements may be WiFi RSS [26], GNSS measurements [131] or human-reported markers [129]. These options are not explored in this thesis.

1.1.2 Map-aided Pedestrian Dead Reckoning

Many researchers are working on the difficult problem of calculating PDR from sensors at different body placements, such as pocket [149][110], wrist [120], foot-mounted [113][166] or hand-held [24][151]. Usually, the placement of the array of sensors in the pedestrian's body determines the type of algorithm that we can apply to infer position and orientation change. For example, if the sensor array is foot-mounted one can perform strap-down inertial navigation [52]; if we use the sensors of a smartphone placed in the pocket, one can resort to step counting, step length estimation and attitude estimation [110].

The challenge of PDR systems based only on inertial sensors is that the measurement error (**drift**) grows unboundedly over time [165]. This is due to small errors in the acceleration and angular velocity measurements, which are integrated into larger errors over time.

Despite the inherent drift of any form of PDR and even in the absence of other sensors, knowledge of the map of the environment can be sufficient to provide accurate long-term pedestrian positioning in 2D and 3D [97][164][14][98][89]. This is because suitable estimators, such a particle filter [137], use the constraints imposed by the knowledge of any relatively stable feature that influences human motion: In addition to walls, this includes many other features of our rich indoor environment such as stalls, large pieces of furniture, displays, counters, shopping aisles, door frames, that are very prominent in many public spaces. However, note that building plans or indoor maps do not usually contain any of this semi-static information and are often proprietary, outdated and rarely available to the public.

1.2 Maps for Indoor Pedestrian Navigation

Several efforts are addressing the creation of a database of indoor maps: Google leads one of the largest efforts to collect indoor building plans [65]; Open Street Maps [114] is also starting to incorporate more detailed indoor plans. However, the available indoor maps rarely reflect the physical layout of the venue with its many obstacles that significantly constrain pedestrian motion.

Even if an existing indoor plan shows the physical layout of an indoor venue, this may undergo changes over time. For example, the physical configuration of a supermarket or

a conference room may change to adapt to new products or events. Sometimes, these changes may happen so rapidly that the existing map can quickly become outdated, obsolete and inaccurate.

We anticipate that collaborative mapping [141][58][122] will play an important role in providing accurate maps and location information that can be used in particular by multimodal travelers [106][125]. To generate such maps in an automated manner, we can resort to the **SLAM** principle [145][45]. Simultaneous Localization and Mapping (SLAM) addresses the estimation of the pose of a moving entity within an unknown environment by simultaneously estimating the map of the environment as it moves.

Currently, many existing cleaning robots commence a cleaning cycle with no memory of previous operations and perform SLAM during the cleaning process. This is a robust and successful approach in most domestic environments where the robot is expected to clean all areas within a room or home. It is also the case of firefighters during a rescue mission in a building.

But for a pedestrian entering a large building who is searching for a specific place, it is more appropriate to draw on an existing map. SLAM in this case might be used to generate the initial map and then only to update the map to account for changes in the environment or to refine its accuracy. In this thesis, we will refer to SLAM as a problem statement pertaining to an algorithmic setting, not an actual application. In other words, a user contributing to a collaborative mapping process might or might not be simultaneously using a positioning service. However, the data collected in such a way will usually be processed by algorithms falling within the SLAM domain. Furthermore, SLAM might be conducted in an offline fashion many hours after the data has been collected.

1.2.1 Robotic SLAM

SLAM [45] was originally presented within the robotics community as a technique to jointly estimate the position of a robot and the map of the detected landmarks. The most well-known algorithmic implementations are: SLAM based on the Extended Kalman Filter (EKF) [40][146][145][100], GraphSLAM [157] and FastSLAM [108], which is based on a Rao-Blackwellized Particle Filter (RBPF) [42].

State-of-the art techniques for robotic mapping rely on cameras [38][88] and range finders (for example laser scanners) [154]. In [153] Thrun surveys different robotic mapping algorithms such as Kalman filter techniques, the expectation maximization algorithm and hybrid approaches (eg. particle filters [137]). He differentiates between occupancy grid maps and object maps.

More recently, a number of approaches have addressed the problem of collaboratively mapping an environment using two or more robots, a problem known as collaborative or (cooperative) SLAM. We can distinguish between techniques where the initial relative position of the robots is well-known [154], techniques that require a rendezvous between

the robots [74] and techniques where the relative initial position of the robots is unknown [95][28][29] (see also survey in [6]). Most collaborative approaches are distributed [152][95][19][27][116][39] to allow for collaborative (online) exploration. The approaches in [19] and [28] combine the maps in a post-processing step in a centralized manner and can be later used for localization.

1.2.2 Pedestrian SLAM

The main difference between robotic and pedestrian SLAM is that in the latter we have no direct control over the pedestrian's path: Whereas in the case of robots we can direct their motion via control inputs, pedestrians roam rooms and other accessible areas of a building following their own will. In addition, the sensors that we might be able to install on a robot platform such as a laser scanner might be too invasive for a pedestrian. Ideally, we would like to perform pedestrian SLAM using the sensors of a smartphone, which are widely spread.

Pedestrian mapping approaches become scalable by **crowdsourcing** [75] the mapping task: a group of pedestrians that carry the necessary sensors collaborate to build a more accurate and complete map of an environment.

Next, we present an overview of existing techniques that address pedestrian SLAM, differentiating between infrastructure-based and infrastructure-free approaches (Table 1.1).

Infrastructure-based Approaches

Most state-of-the-art smartphone-based approaches to pedestrian SLAM assume the existence of a pre-installed infrastructure, for example WiFi access points [26][50][141]. WiFiSLAM [50] uses Gaussian process latent variable models and a motion dynamics model to estimate the location of unlabeled signal strength data. WiSLAM [26] maps the position of the detected WiFi access points as the pedestrian walks.

SmartSLAM [143] uses the measurements of the inertial sensors of a smartphone (gyroscopes, accelerometers) for the measurement model and WiFi fingerprints for the observation model. CrowdInside [2] goes a step further and uses GPS traces and detected elevators, stairways and escalators as anchor points to reset the errors of the noisy sensors of a smartphone. Furthermore it is able to reconstruct the floor plan outline.

CrowdInside [2], SignalSLAM [107] and Walkie-Markie [141] rely on WiFi RSS measurements (in the case of SignalSLAM also on any available infrastructure such as Bluetooth, LTE, etc.) to correct the drifting errors of the step estimation algorithms performed in the smartphone using its inertial sensors. Once the individual traces have been generated, they are merged in a centralized manner.

There are also hybrid solutions in which humans and robots collaborate, such as [92], which requires RFID tags to be previously deployed.

Infrastructure-free Approaches

These approaches rely uniquely on sensors — IMUs, cameras, laser scanners, the sensors of a smartphone, etc — that the pedestrian wears or carries while walking, for example mounted on the shoe [128], in the pocket [86], on the chest [48], on the wrists [69], on the head [32], installed on a backpack [66], in the form of a suit [67] or carrying them in the hand [35][64].

Depending on the sensors used, different landmarks or features can be *slammed*[2)]: ActionSLAM [69][70] and and the approach in [67] map location-related activities such as door handling, sitting, drinking, writing, reading, etc. In [48] the authors rely on cameras and LIDAR to map landmarks in the environment. FootSLAM [8] and FeetSLAM [136] build probabilistic maps of human motion using only pedestrian step measurements.

1.2.3 FootSLAM and FeetSLAM in a Nutshell

FootSLAM [128] is an infrastructure-independent technique that can estimate **maps of human step direction probabilities** (also called probabilistic maps of human motion) in constrained environments (Fig. 1.1).

Figure 1.1: Walls, doors, obstacles and other constraints in the environment channel human motion.

A pedestrian walks collecting measurements of his steps with an accelerometer, a gyroscope and a magnetometer. These sensors are imperfect and their measurements are inaccurate. To estimate and correct the sensor errors, FootSLAM builds on a **cascaded estimation architecture** that comprises: 1) a low-level Kalman Filter [162] that fuses high rate measurements collected by the sensors to produce (still noisy) step measurements (odometry), and 2) an upper-level PF [137] that tracks different odometry error

[2)] Slamming here denotes the process of applying SLAM, in contrast to mapping

	Technique	Cit.	Sensors	Summary
Infrastructure-based Approaches	WiSLAM	[26]	WiFi radio, IMU	Step measurements are fused with RSS measurements following a FastSLAM approach to map the position of the detected Wi-Fi access points.
	WiFiSLAM	[50]	WiFi radio	Uses Gaussian process latent variable models and a motion dynamics model to estimate the location of unlabeled signal strength data.
	SmartSLAM	[143]	GPS, WiFi radio, accelerometer, gyroscope, magnetometer	Switches between different sensor fusion algorithms based on EKF and Particle Filter (PF) depending on the level of certainty in the system and on the availability of a WiFi fingerprint map.
	CrowdInside	[2]	GPS, WiFi radio, accelerometer, gyroscope, magnetometer	Uses GPS traces and detected elevators, stairways and escalators as anchor points to reset the errors of the noisy sensors of a smartphone and to reconstruct the floor plan outline.
	RFID-SLAM	[92]	RFID tags, IMU	Corrects odometry trajectories from multiple collaborating robots and humans while utilizing RFID transponders for data association.
	SignalSLAM	[107]	WiFi, Bluetooth, LTE radio, accelerometer, gyroscope, magnetometer	Builds multi-modal RF signal maps using the GraphSLAM principle.
	WalkieMarkie	[141]	WiFi radio, accelerometer, gyroscope, magnetometer	Creates internal pathway maps of buildings using WiFi-Marks (trends in the WiFi signal strength) as landmarks.
Infrastructure-free Approaches	HeadSLAM	[32]	Laser, IMU	Uses a Rao-Blakwellized Particle Filter to build a map of landmarks of the environment. Scan matching is used to improve the odometry-based position estimate.
	ActionSLAM	[69]	Three IMUs	Relies on a foot-mounted IMU for positioning and a hip-worn sensor for both activity recognition and computation. Builds on FastSLAM to estimate a map of location-tied activities.
	XSens MVN suit	[67]	IMUs built on a suit	Corrects odometry drift by using human activities as landmarks as a pedestrian moves and constructs a topological map of the visited areas.
	Project Tango	[64]	Camera, Infrared (IR), accelerometer, gyroscope, magnetometer	Tracks the location of the user using visual-inertial odometry and corrects the odometry drift using the learned map of the visited area, composed of the key visual features of a physical space.
	Project Cartographer	[66]	Laser scanner, IMU	Maps the indoor environment in real time and displays it on an Android tablet connected to the backpack's computer.
	Multi-floor SLAM	[48]	Camera, LIDAR, IMU, barometer	Fuses multiple sensor measurements following an incremental smoothing and mapping algorithm to map multifloor environments.
	FootSLAM, PocketSLAM	[8][86]	IMU	Built on a FastSLAM approach to estimate the location of the pedestrian and learn the map of walkable areas by processing pedestrian step measurements.
	FeetSLAM	[136]	IMU	Processes multiple odometry datasets iteratively with FootSLAM and merges them by finding the geometric transformation between the individual maps.

Table 1.1: Overview of pedestrian SLAM techniques.

hypotheses. When the pedestrian revisits an area (what we call **"closing the loop"**), the errors can be corrected and a map of the visited areas is generated.

But a pedestrian who casually roams the shops, aisles and hallways at a mall or an airport might not visit all areas in the building and might not "close the loop". To deal

with this, the collaborative counterpart of FootSLAM, called FeetSLAM [136], uses other pedestrian's steps to close the loop. The idea behind this is that walls, doors, furniture and other obstacles constrain different pedestrians to walk following similar patterns (Fig. 1.1).

Operating in an iterative fashion, the "Turbo" FeetSLAM algorithm has been able to merge several datasets collected by one or more pedestrians while walking within the same environment, creating an extensive and more accurate probabilistic map of the visited areas. One of the requirements is that the relative pose of the pedestrians is more or less known [136].

1.3 Goals and Research Questions

FootSLAM [128] and its collaborative counterpart FeetSLAM [136] address the problem of crowdsourcing the generation of pedestrian motion maps that can be used either in an online or offline fashion for improved indoor pedestrian navigation. The goal is to provide users with with sub-meter accurate maps to improve customer experiences in airports and shopping centers, reduce the time needed to find victims after an emergency call or coordinate a firefighter team during a rescue mission.

The aim of this work is to tackle the main deficiencies of FootSLAM and FeetSLAM to bridge the gap towards fully automated crowdsourced indoor mapping. The most relevant shortcomings of FootSLAM and FeetSLAM that are tackled in this thesis are:

1. **Single Floor Mapping:** The mathematical foundation and implementation of FootSLAM is based solely on single floor exploration and does not allow us to map multistory buildings. This thesis extends FootSLAM and the combination of maps to multistory environments.

2. **Offline Processing:** FootSLAM's naïve implementation of FastSLAM exhibits a computational complexity that grows linearly with the area visited by the pedestrian, allowing only offline processing of collected data. This thesis proposes an algorithm to lower the computational complexity of FootSLAM with the visited area, to allow mapping large areas in quasi real-time.

3. **Slow Map Combination:** FeetSLAM combines a set of individual maps by finding the geometric transformation that places each pair of individual maps within the same coordinate system and combining first those maps that best explain the observed spatial transitions of each other. This grows quadratically in the number of maps. In addition, each combination of two maps grows quadratically in the area explored by the user, making the algorithm very time-costly. To mitigate these problems, in this thesis we will:

    a) Define a metric that reflects the *uncertainty* of a FootSLAM map, i.e. how accurate or complete a map is. This metric will allow us to sort a set of maps by quality, which can be used to select the order in which the maps

can be combined. This reduces the computational complexity growth rate of combining the maps from quadratic to linear in the number of maps.

b) Reduce the computational complexity of combining two maps from quadratic to linear in the area explored using a novel approach based on the Hough transform [44].

4. **Few Collected Datasets:** So far FootSLAM and FeetSLAM have been evaluated by processing datasets collected within two different buildings. During the development of this thesis, we have collected numerous additional datasets at other locations that further validate the FootSLAM and FeetSLAM algorithms.

5. **Lack of Large-scale Mapping Analysis:** Previous work has never analyzed the costs of mapping the entire indoor world. This thesis will study the time needed to map an area (by analyzing the frequency of visits of different building types) and analyze the requirements to create and store indoor maps for the entire indoor world.

1.4 Thesis Structure

This thesis is structured as follows:

Chapter 2 provides technical background on Bayes filters — including the Kalman Filter (KF) and the PF —, SLAM, collecting and processing human odometry measurements and the principles of FootSLAM and FeetSLAM.

Chapter 3 introduces a number of metrics based on (information) entropy that reflect the uncertainty of a FootSLAM map and the uncertainty of a sequence of pedestrian's steps in an environment. The contents of this chapter have appeared in [84].

Chapter 4 extends FootSLAM to allow mapping multistory environments. The contents of this chapter have appeared in [56].

Chapter 5 introduces a new data structure in FootSLAM that effectively reduces its computational complexity and memory requirements and allows for quasi-real time processing. The contents of this chapter have appeared in [60], [61] and [56].

Chapter 6 proposes a new approach to quickly combine different FootSLAM maps, which opens the door to online cooperative mapping applications. The contents of this chapter have been submitted to a journal [59].

Chapter 7 studies the requirements to build pedestrian motion maps of the entire indoor world and describes the applications of FootSLAM probabilistic motion maps as well as deployment alternatives for FeetSLAM. The contents of this chapter have appeared in [58] and in [57].

Chapter 8 outlines the conclusions and possible future lines of work.

2 Technical Background

This chapter provides an introduction to the technical background that will help the reader get familiarized with the terms used in the following chapters.

FootSLAM is built on a two-tier **cascaded estimation architecture** [97]. The components are:

1. A lower level human odometry generator that fuses the high-rate inertial measurements (of the order of hundreds of Hz) to estimate the position and direction change of the pedestrian, for example based on the KF (see Sections 2.2.1 to 2.2.3). Its output is a Gaussian distribution representing the length and direction change of the last step (see Section 2.4).

2. An upper level PF (details in Section 2.2.4) that runs at a lower rate (roughly at 1 Hz) and fuses the step measurements provided by the odometry generator with any other available measurements. This is done following the FastSLAM factorization (see Section 2.3.4).

FootSLAM is an implementation of the upper level PF that relies on repeatedly observing similar spatial transitions (when the pedestrian revisits an area) to correct the errors that perturb the odometry measurements.

The principle of this cascaded architecture is applicable to any form of odometry measurements, for example those collected with the sensors of a smartphone or other mobile device placed in the pocket, attached to the wrist or hand-held (see Section 2.4). Although all odometry datasets in this thesis have been collected with a foot-mounted IMU, FootSLAM is agnostic to the source of the odometry; the only requirement is that measurement likelihoods used in the filter are modeled according to the new odometry source (e.g. PocketSLAM [86]).

This chapter starts with an introduction to big O notation (Section 2.1) followed by an overview of the theory behind the Bayes filters used in this thesis (Section 2.2). Next, we provide an introduction to SLAM (Section 2.3) and we show how to collect human odometry measurements (Section 2.4). Finally, we describe the basis of FootSLAM (Section 2.5) and its multiuser extension FeetSLAM (Section 2.6), which are the cornerstones of this thesis.

2.1 Big O Notation

In this thesis we will analyze the performance of many algorithms. An important aspect of analyzing an algorithm refers to the determination of the needed resources for its execution (computational time or memory requirements) as a function of the algorithm's input size [33].

Algorithms are usually analyzed in the asymptotic sense, i.e. for an arbitrary large input size. Big O notation, omega notation and theta notation are used to this purpose. Here we focus on big O notation: Big O notation, usually denoted as $\mathcal{O}$, is used to express the worst-case scenario for a given algorithm by providing an upper bound for its growth rate. An algorithm exhibits a growth rate on the order of a mathematical function $\mathcal{O}(f(n))$ if [33]:

$$0 \leq T(n) \leq C \cdot f(n); \forall\, n \geq n_0, \tag{2.1}$$

where C and n_0 are positive constants.

An algorithm is said to be constant growth rate if its complexity growth rate is bounded by a value that does not depend on the size of the input n. It is written as $\mathcal{O}(1)$.

2.2 Bayes Filters

Many scientific problems require estimating the state of a dynamic system given noisy measurements (also called observations) as they become available, i.e. processing them sequentially rather than in batch. For instance, in pedestrian navigation we are interested in estimating the pose of a pedestrian given (noisy) measurements collected as he walks, for example odometry measurements, GNSS measurements, etc.

Bayes filters provide a suitable framework to address this. A Bayes filter, also known as recursive Bayesian estimator, is a general probabilistic method that recursively estimates the posterior **probability density function (pdf)** of the system over time, as new incoming measurements become available, and taking into account the full state of the system including any prior knowledge about the system dynamics. An estimator that uses all available information is called optimal [155].

In Bayes filters, the true state at discrete time stamp $k \in \mathbb{N}$ is a random variable encoded by $\boldsymbol{x}_k$, which is assumed to be a "hidden" Markov process. For our application, we assume this process to be a Markov process of first order, i.e. the future state $\boldsymbol{x}_k$ is conditionally independent from past states $\boldsymbol{x}_{0:k-2}$ given the the present state $\boldsymbol{x}_{k-1}$. Mathematically:

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}), \tag{2.2}$$

where $\boldsymbol{x}_{0:k-1}$ is the history of states up to time instant $k-1$.

The goal of a Bayes filter is to estimate the full posterior:

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k}), \tag{2.3}$$

where $\boldsymbol{z}_{1:k}$ is the history of noisy measurements, also called observations (e.g. laser scans, pedestrian steps, etc.). See Dynamic Bayesian Network (DBN) in Fig. 2.1.

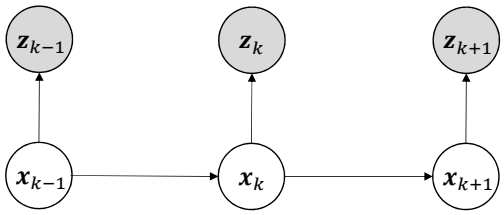


Figure 2.1: DBN that characterizes the evolution of the "hidden" Markov process of first order $\boldsymbol{x}$ (state) and the observations $\boldsymbol{z}$.

The process is usually defined quantitatively with two functions [137][155]:

1. A possibly nonlinear function of the state transition $f_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$:

$$\boldsymbol{x}_k = f_k(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1}), \tag{2.4}$$

   with $\boldsymbol{v}_{k-1}$ an i.i.d. process noise sequence and n_x, n_v the dimensions of the state and process noise vectors, respectively.

2. A possibly nonlinear measurement function $h_k : \mathbb{R}^{n_z} \times \mathbb{R}^{n_n} \rightarrow \mathbb{R}^{n_x}$:

$$\boldsymbol{z}_k = h_k(\boldsymbol{x}_k, \boldsymbol{n}_k), \tag{2.5}$$

   with $\boldsymbol{n}_k$ an i.i.d. measurement noise sequence and n_z, n_n the dimensions of the measurement and measurement noise vectors, respectively.

Then, the estimation task is divided into two stages that are repeated every time a new measurement becomes available [137]:

1. Prediction step: Uses the system model encoded in (2.4) to obtain the prior pdf of $\boldsymbol{x}_k$:

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})\ p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})\ \mathrm{d}\boldsymbol{x}_{k-1}. \tag{2.6}$$

   Note that the second term in the integral is the recursive term, which is assumed known at time stamp k. In practice, it is assumed that an initial estimate of $p(\boldsymbol{x}_0|\boldsymbol{z}_0) = p(\boldsymbol{x}_0)$ is available. In addition, here we have applied that the state at time stamp k is conditionally independent of the history of observations $\boldsymbol{z}_{1:k}$ conditioned on the previous state $\boldsymbol{x}_{k-1}$: $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{z}_{1:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ (see DBN in Fig. 2.1).

2. Update step: The newly available measurement $\boldsymbol{z}_k$ is used to update the prior density and compute the posterior using the Bayes' rule:

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)\ p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})}, \tag{2.7}$$

where the denominator is a normalizing constant:

$$p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1}) = \int p(\boldsymbol{z}_k|\boldsymbol{x}_k)\ p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})\ \mathrm{d}\boldsymbol{x}_k, \tag{2.8}$$

and $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ is the likelihood function defined by the measurement model (2.5).

The prediction and update equations (2.6) and (2.7) are a conceptual solution for the estimation problem that cannot always be solved analytically. The two main families of state estimation derived from the Bayes filter are Gaussian filters and non-parametric filters.

Gaussian filters (e.g. KF, EKF, Unscented Kalman Filter (UKF) and information filter) represent the posteriors by multivariate normal distributions and are the first tractable solution to the estimation process. Non-parametric filters (e.g. histogram filters, particle filters) approximate the posterior with a finite number of parameters and are suitable for multivariate data and nonlinear/non-Gaussian processes with non-tractable solutions.

The following subsections provide a short introduction to the KF, the EKF and the UKF, which are the basis to process inertial sensor measurements, followed by an introduction to particle filters, with special focus on Rao-Blackwellized particle filters (which is used in FootSLAM).

2.2.1 The Kalman Filter (KF)

The KF is a Gaussian filter that represents the posterior using a multivariate normal distribution. The well-known formula for a multivariate normal distribution with mean vector μ and covariance matrix Σ is:

$$\mathcal{N}(\boldsymbol{x};\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}\det(\Sigma)^{1/2}} \exp\{-\frac{1}{2}(\boldsymbol{x}-\mu)^T\ \Sigma^{-1}(\boldsymbol{x}-\mu)\}, \tag{2.9}$$

where $\det(\cdot)$ represents the determinant of the matrix in brackets.

If $p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})$ is Gaussian, it can be shown that $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ is also Gaussian if the following requirements are met [162][137]:

1. The initial posterior $p(\boldsymbol{x}_0)$ must be normally distributed.
2. The state transition function $f_k(\boldsymbol{x}_{k-1}, \boldsymbol{v}_{k-1})$ must be a linear function in its arguments with added Gaussian noise $\boldsymbol{v}_{k-1}$ of covariance Q_{k-1}:

   $$\boldsymbol{x}_k = F_k\boldsymbol{x}_{k-1} + \boldsymbol{v}_{k-1}, \tag{2.10}$$

   where F_k is a known matrix that defines the linear function between $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$.
3. The measurement function $h_k(\boldsymbol{x}_k, \boldsymbol{n}_k)$ must be linear in its arguments with added Gaussian noise $\boldsymbol{n}_k$ of covariance R_k:

   $$\boldsymbol{z}_k = H_k\boldsymbol{x}_k + \boldsymbol{n}_k, \tag{2.11}$$

   where H_k is a known matrix that defines the linear function between $\boldsymbol{x}_k$ and $\boldsymbol{z}_k$.

Applying these assumptions to equations (2.6) and (2.7) yields:

$$
\begin{aligned}
p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) &= \mathcal{N}(\boldsymbol{x}_{k-1}; \mu_{k-1|k-1}, \Sigma_{k-1|k-1}) && (2.12)\\
p(\boldsymbol{x}_{k}|\boldsymbol{z}_{1:k-1}) &= \mathcal{N}(\boldsymbol{x}_{k}; \mu_{k|k-1}, \Sigma_{k|k-1}) && (2.13)\\
p(\boldsymbol{x}_{k}|\boldsymbol{z}_{1:k}) &= \mathcal{N}(\boldsymbol{x}_{k}; \mu_{k|k}, \Sigma_{k|k}), && (2.14)
\end{aligned}
$$

with:

$$
\begin{aligned}
\mu_{k|k-1} &= F_k \mu_{k-1|k-1} && (2.15)\\
\Sigma_{k|k-1} &= Q_{k-1} + F_k \Sigma_{k-1|k-1} F_k^T && (2.16)\\
\mu_{k|k} &= \mu_{k|k-1} + K_k(\boldsymbol{z}_k - H_k \mu_{k|k-1}) && (2.17)\\
\Sigma_{k|k} &= \Sigma_{k|k-1} - K_k H_k \Sigma_{k|k-1}, && (2.18)
\end{aligned}
$$

and:

$$
\begin{aligned}
K_k &= \Sigma_{k|k-1} H_k^T S_k^{-1} && (2.19)\\
S_k &= H_k \Sigma_{k|k-1} H_k^T + R_k, && (2.20)
\end{aligned}
$$

where K_k is the Kalman gain and S_k specifies the covariance of the innovation term $(\boldsymbol{z}_k - H_k \mu_{k|k-1})$ [137]. The Kalman gain specifies the impact of the new measurement in the new posterior estimate.

The KF is computationally efficient and it is optimal in the Bayesian sense if the aforementioned assumptions hold. However, KFs deal poorly with non-linearities in the state transition and measurement models. To deal with Gaussian systems with non-linearities, the EKF and UKF are called into play, which are suboptimal algorithms.

2.2.2 The Extended Kalman Filter (EKF)

The EKF relaxes the assumption that the state transition function and the measurement function must be linear in their parameters. The key idea is to approximate the non-linear functions by a linear function that is tangent to it at the mean of the Gaussian [162]. EKFs rely on first order Taylor expansion for this linearization and approximate the term $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ by a Gaussian, yielding:

$$
\begin{aligned}
p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) &\approx \mathcal{N}(\boldsymbol{x}_{k-1}; \mu_{k-1|k-1}, \Sigma_{k-1|k-1}) && (2.21)\\
p(\boldsymbol{x}_{k}|\boldsymbol{z}_{1:k-1}) &\approx \mathcal{N}(\boldsymbol{x}_{k}; \mu_{k|k-1}, \Sigma_{k|k-1}) && (2.22)\\
p(\boldsymbol{x}_{k}|\boldsymbol{z}_{1:k}) &\approx \mathcal{N}(\boldsymbol{x}_{k}; \mu_{k|k}, \Sigma_{k|k}), && (2.23)
\end{aligned}
$$

with:

$$\mu_{k|k-1} = f_k(\mu_{k-1|k-1}) \tag{2.24}$$
$$\Sigma_{k|k-1} = Q_{k-1} + \hat{F}_k \Sigma_{k-1|k-1} \hat{F}_k^T \tag{2.25}$$
$$\mu_{k|k} = \mu_{k|k-1} + K_k(\boldsymbol{z}_k - h_k(\mu_{k|k-1})) \tag{2.26}$$
$$\Sigma_{k|k} = \Sigma_{k|k-1} - K_k \hat{H}_k \Sigma_{k|k-1}, \tag{2.27}$$

where $f_k(\cdot)$ and $h_k(\cdot)$ are now non linear functions and $\hat{F}_k$ and $\hat{H}_k$ are local linearizations of those non-linear functions using the first order Taylor expansion [162].

The main drawback of the EKF is that the linearization of the functions can yield highly unstable filters when the local linearity assumption does not hold. Moreover, the derivation of the Jacobian matrices, $\hat{F}_k$ and $\hat{H}_k$, is often problematic and can lead to implementation difficulties. For Gaussian systems with stark non-linearities, we can turn to Unscented Kalman Filters.

2.2.3 The Uncented Kalman Filter (UKF)

The UKF applies a different linearization via the so-called unscented transform. The idea is to deterministically extract *sigma points* from the Gaussian and pass these through $f(\cdot)$. These sigma points are chosen as follows: one located at the mean and two per dimension of the covariance, symmetrically placed along the main axis of each dimension [81]. The reader can find the mathematical derivation in [81] and [155].

For non-linear systems the UKF performs equal to or better than the EKF, depending on the degree of non-linearity and the uncertainty of the prior. An advantage of the UKF over the EKF is that it does not require to compute Jacobians, which in many cases are difficult to derive [81].

Note that if the true probability distribution $p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k})$ is non-Gaussian then a Gaussian-based approximation will never describe it well. For non-Gaussian/non-linear systems, we can resort to particle filters.

2.2.4 Particle Filter (PF)

A particle filter is a sequential Monte Carlo algorithm that implements a recursive Bayes filter by Monte Carlo simulations. The goal is to sequentially estimate a desired posterior pdf by a set of N_p random samples, called particles, each associated with a weight that represents the probability of the particle being sampled from the pdf. As the number of particles increases, the particle filter approaches the optimal Bayesian estimator [41].

The mathematical formulation for this discrete representation of the full pdf is:

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k}) \approx \sum_{m=0}^{N_p-1} w_k^{[m]} \delta(\boldsymbol{x}_{0:k} - \boldsymbol{x}_{0:k}^{[m]}), \tag{2.28}$$

where $\delta(\cdot)$ is the Dirac delta, $\boldsymbol{x}_{0:k}$ is the history of unobservable states whose pdf we would like to estimate, $\boldsymbol{z}_{1:k}$ is the history of observable states (e.g. measurements) and $w_k^{[m]}$ represents the weight assigned to particle m. The higher the density, the higher the weight (Fig. 2.2). The weights are normalized so that $\sum_{m=0}^{N_p-1} w_k^{[m]} = 1$.

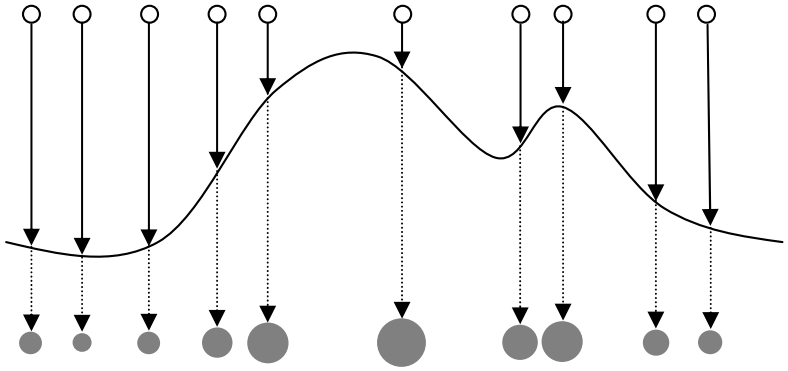


Figure 2.2: Particle filter representation of a posterior density function. The circles in the upper part represent the particles (samples). The circles in the lower part represent the weight assigned to each particle: the higher the density, the higher the weight.

Sampling from the state posterior $p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})$ at any given point in time k is seldom possible. The most common approach to overcome this is the Sequential Importance Sampling (SIS) algorithm.

2.2.5 Sequential Importance Sampling (SIS) Algorithm

The SIS algorithm is the basis of most particle filters. The key idea is to choose the weights of the particles using the principle of **importance sampling** [41].

Assuming that $p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k}) \propto q(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})$ is a pdf from which it is impossible to draw samples but for which $q(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})$ can be evaluated, we can define $q(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})$ as the *importance density*. If we draw N_p particles $\boldsymbol{x}_{0:k}^{[m]}$ (2.28) from $q(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})$, then the importance weights are defined as [137]:

$$w_k^{[m]} \propto \frac{p(\boldsymbol{x}_{0:k}^{[m]}|\boldsymbol{z}_{1:k})}{q(\boldsymbol{x}_{0:k}^{[m]}|\boldsymbol{z}_{1:k})}. \tag{2.29}$$

For this principle to work in a sequential manner, we can choose the importance density to be factorized as follows [137]:

$$q(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k}) = q(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{z}_{1:k}) \cdot q(\boldsymbol{x}_{0:k-1}|\boldsymbol{z}_{1:k-1}) = q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{z}_k) \cdot q(\boldsymbol{x}_{0:k-1}|\boldsymbol{z}_{1:k-1}), \tag{2.30}$$

and after introducing (2.30) in (2.29), the weight update equation becomes [137]:

$$w_k^{[m]} \propto w_{k-1}^{[m]} \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k^{[m]})p(\boldsymbol{x}_k^{[m]}|\boldsymbol{x}_{k-1}^{[m]})}{q(\boldsymbol{x}_k^{[m]}|\boldsymbol{x}_{k-1}^{[m]}, \boldsymbol{z}_k)}. \tag{2.31}$$

The posterior pdf at time stamp k is then approximated by:

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \approx \sum_{m=0}^{N_p-1} w_k^{[m]} \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^{[m]}), \tag{2.32}$$

with the importance weights defined in (2.31)

A well-known problem of the SIS particle filter is the **degeneracy problem or particle depletion**, whereby the unconditional variance of the importance weights increases over time. The degeneracy problem arises after a few iterations, when all particles except for one have low weight, hence hardly contributing to the computation of the posterior [41]. Two methods are well-known to mitigate this problem: choosing an appropriate importance density function and resampling.

Importance Density Function Choice

Choosing a suitable importance density function is crucial for the performance of the particle filter. In [137], a number of importance density functions are proposed. The simplest proposal distribution is to sample from the transitional prior:

$$q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]}, \boldsymbol{z}_k) \triangleq p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]}), \tag{2.33}$$

which simplifies the weight update (2.31) to:

$$w_k^{[m]} \propto w_{k-1}^{[m]}\ p(\boldsymbol{z}_k|\boldsymbol{x}_k^{[m]}). \tag{2.34}$$

However, there are cases in which the sensor measurements $\boldsymbol{z}_{0:k}$ are more informative than the motion model, and the resulting likelihood $p(\boldsymbol{z}_k|\boldsymbol{x}_k^{[m]})$ has a much narrower variance compared to the prior. Accordingly the posterior is closer in similarity to the likelihood than to the prior and it has been shown that looking at the measurement $\boldsymbol{z}_k$ improves performance [137][111]:

$$q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]}, \boldsymbol{z}_k) \triangleq p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]}, \boldsymbol{z}_k) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k)\ p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]})}{p(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}^{[m]})}, \tag{2.35}$$

yielding for the weight update (2.31):

$$w_k^{[m]} \propto w_{k-1}^{[m]}\ p(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}^{[m]}) = w_{k-1}^{[m]} \int p(\boldsymbol{z}_k|\boldsymbol{x}_k)p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]})\ \mathrm{d}\boldsymbol{x}_k \tag{2.36}$$

Such importance density for the proposal step is optimal given that for any $\boldsymbol{x}_{k-1}^{[m]}$, the new weight $w_k^{[m]}$ is conditionally independent of the value drawn for $\boldsymbol{x}_k^{[m]}$ conditioned on the previous value $\boldsymbol{x}_{k-1}$.

In general we cannot analytically sample from $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]}, \boldsymbol{z}_k)$ or evaluate the integral that computes the predictive density $p(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}^{[m]})$. However, when $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^{[m]}, \boldsymbol{z}_k)$ is Gaussian, we can sample from it.

Resampling

Resampling means drawing N_p' new particles from the existing set of N_p particles (usually $N_p' = N_p$) according to their importance weight $w_k^{[m]}$ (the higher the weight, the larger the number of particles that are drawn from it) [45]. Drawing a particle (called "resampling" particle) from a generating or "parent" particle usually means copying all its state information with replacement. Resampling allows the particle filter to concentrate on particles with large weight and eventually eliminate those that have a low weight.

Resampling can be performed deterministically, i.e. every n steps, or dynamically, when the following approximation of the effective number of particles N_{eff} [137]:

$$N_{\text{eff}} = \frac{1}{\sum_{m=0}^{N_p-1}(w_k^{[m]})^2}, \tag{2.37}$$

falls under a threshold.

Particle filters based on the SIS algorithm pose the disadvantage that sampling in high-dimensional spaces can be inefficient, with the computational complexity growing exponentially with the dimension of the explored state space. Using the Rao-Blackwell formula, we can marginalize out some of the variables for which a "tractable substructure" exists [42].

2.2.6 The Rao-Blackwellized Particle Filter (RBPF)

Suppose we can split the hidden variables $\boldsymbol{x}_{0:k}$ into two groups: $\boldsymbol{x}_{0:k} = \{\boldsymbol{r}_{0:k}, \boldsymbol{y}_{0:k}\}$, with $p(\boldsymbol{y}_{0:k}|\boldsymbol{r}_{0:k}, \boldsymbol{z}_{1:k})$ being analytically tractable with a parametric representation. Then the full posterior $p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k})$ becomes:

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{z}_{1:k}) = p(\boldsymbol{r}_{0:k}, \boldsymbol{y}_{0:k}|\boldsymbol{z}_{1:k}) = p(\boldsymbol{y}_{0:k}|\boldsymbol{r}_{0:k}, \boldsymbol{z}_{1:k})\ p(\boldsymbol{r}_{0:k}|\boldsymbol{z}_{1:k}). \tag{2.38}$$

Then, for each particle we store, not only a sample from $p(\boldsymbol{r}_{0:k}|\boldsymbol{z}_{1:k})$, represented by $\boldsymbol{r}_{0:k}^{[m]}$, but also a parametric representation for $p(\boldsymbol{y}_{0:k}|\boldsymbol{r}_{0:k}^{[m]}, \boldsymbol{z}_{1:k})$.

Since the dimension of $p(\boldsymbol{r}_{0:k}|\boldsymbol{z}_{1:k})$ is lower than that of $p(\boldsymbol{r}_{0:k}, \boldsymbol{y}_{0:k}|\boldsymbol{z}_{1:k})$, then the dimensionality of the space in which we sample is reduced, the computational complexity

is lowered, and the same accuracy is expected with a lower number of particles. This particle filter is called RBPF [42].

As we will show in Section 2.5, the RBPF is the basis of the FastSLAM factorization, used in FootSLAM.

2.3 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) addresses the problem of placing a moving subject (e.g. a robot carrying laser scanners) in an unknown environment to simultaneously build a map of the environment and determine the pose of the subject within that map [45][12].

Localizing a moving subject within a known map (**localization**) or building a map of the environment given a series of known poses (**mapping**) are less complex problems in that now both the map and the poses need to be simultaneously estimated as the subject moves. SLAM can be seen as an instance of the "chicken and egg" problem: What comes first, the map or the pose?

Solutions to the SLAM problem can be divided into *online SLAM problem* and *full SLAM problem*, which we will describe shortly. Before that, we introduce the map variable.

2.3.1 Maps

In the context of SLAM, a map comprises a set of objects in the environment [155]:

$$\boldsymbol{m} = \{\boldsymbol{m}^i\}, \quad 0 \leq i \leq N_m - 1, \tag{2.39}$$

where N_m is the number of objects in the environment and $\boldsymbol{m}^i$ specifies the properties of object i. There are two main types of maps: feature maps and location-based maps. Feature maps comprise landmarks and $\boldsymbol{m}^i$ also contains the coordinates of the landmark. In location-based maps the index i refers to the physical location of the object.

Feature maps are commonly used when mapping an environment, where the map is constructed from exteroceptive sensor data (e.g. laser scanners). Location-based maps such as occupancy grid maps, where each element $\boldsymbol{m}^i$ specifies whether a cell in the grid is occupied or not, are widely used in mobile robot navigation and are suitable for path planning.

A problem that arises when using feature maps is that the landmarks may not be uniquely identifiable, leading to the so-called data association problem: one must infer from the data the correspondence between an observed feature and a landmark. To do this, usually a correspondence variable $\boldsymbol{c}_k^i$ is introduced to denote the correspondence between the feature observed at discrete time k and the landmark $\boldsymbol{m}^i$.

2.3.2 Online SLAM Problem

Online SLAM problems (see DBN in Fig. 2.3) estimate the joint posterior over the state at time instant k, $\boldsymbol{x}_k$, and the (time invariant) map, $\boldsymbol{m}$, conditioned on the history of measurements $\boldsymbol{z}_{1:k}$ and control inputs to the robot $\boldsymbol{u}_{0:k}$:

$$p(\boldsymbol{x}_k, \boldsymbol{m} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k}), \tag{2.40}$$

in a recursive fashion.

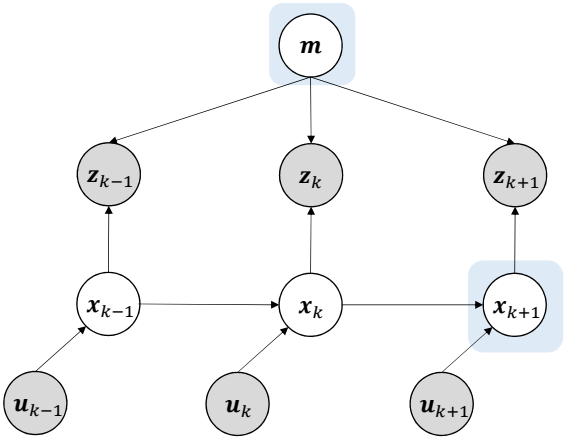


Figure 2.3: DBN representing the online SLAM problem. The goal is to estimate the joint posterior over the current state and the map.

Examples of online SLAM are SLAM based on the EKF (EKF-SLAM) [146][145][100], SLAM with Sparse Extended Information Filters (SEIF-SLAM) [156] and SLAM based on the RBPF – known as FastSLAM [108]. EFK-SLAM and SEIF-SLAM are briefly introduced next, whereas the FastSLAM algorithm is detailed in Section 2.3.4.

EKF-SLAM

EKF-SLAM is the earliest implementation of the SLAM algorithm [155]. EKF-SLAM maps are feature-based maps that comprise point landmarks. The more unique these landmarks are, the better the performance. Since EKF-SLAM is based on the EKF, it relies the Gaussian noise assumption for both the state transition model and measurement model [100]. The idea is to use an incremental maximum likelihood estimator to determine correspondences between the measurements and the landmarks, never revising a correspondence decision [155].

The main limitations of EKF-SLAM is that updates require time quadratic in the number of landmarks in the map and landmarks need to be sufficiently unambiguous [155].

SEIF-SLAM

The SEIF-SLAM algorithm represents the posterior by a sparse information matrix. This renders both the measurement step and motion update step independent of the number of landmarks. For data association it can rely on an incremental maximum likelihood estimator or perform a search on a tree containing all data association [156].

The main drawback of this technique is that it is based on a number of approximations to first linearize and then sparsify the information matrix, resulting in less accurate results than EKF-SLAM [155].

2.3.3 Full SLAM Problem

The full SLAM problem (see DBN in Fig. 2.4) addresses the estimation of the joint posterior over the history of the state $\boldsymbol{x}_{0:k}$ and the (time invariant) map, $\boldsymbol{m}$, conditioned on the history of measurements $\boldsymbol{z}_{1:k}$ and control inputs to the robot $\boldsymbol{u}_{0:k}$:

$$p(\boldsymbol{x}_{0:k}, \boldsymbol{m} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k}). \tag{2.41}$$

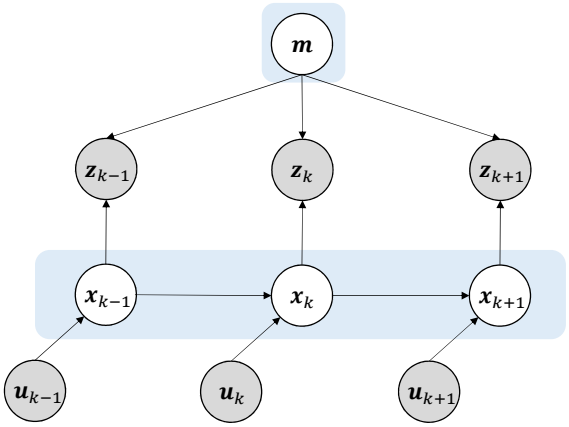


Figure 2.4: DBN representing the full SLAM problem. The goal is to estimate the joint posterior over the whole history of the state and the map.

Two algorithms that address the calculation of the full posterior are GraphSLAM [157] and FastSLAM [108]. Because of its relevance in this thesis, FastSLAM is addressed in the following subsection (Section 2.3.4).

GraphSLAM

The main idea behind GraphSLAM is to represent the full SLAM problem with a sparse graph, which leads to a sum of nonlinear quadratic constraints [157]. The nodes in the

graph are the robot poses $\boldsymbol{x}_{0:k}$ and the map landmarks $\boldsymbol{m} = \{\boldsymbol{m}^i\}$. A *motion arc* joins two consecutive robot poses, whereas a *measurement arc* links the pose with the landmarks that were measured from that pose. Each edge can be seen as soft constraint between poses and landmarks and is represented by the negative log likelihood of the motion or measurement models, respectively.

In GraphSLAM, the set of (nonlinear) contraints are linearized and transformed into a sparse information matrix and an information vector. Optimizing the graph, i.e. minimizing the sum of all constraints, leads to a nonlinear least squares problem and results in a maximum likelihood estimate for the robot path and the map.

The key limitations of GraphSLAM are that the size of the graph grows linearly over time, as the robot moves, and data association requires performing a greedy search algorithm over all data association variables [157].

2.3.4 FastSLAM: Rao-blackwellizing the SLAM Problem

FastSLAM is a particle filter-based implementation of SLAM. As we saw, particle filters are powerful estimators to maintain multiple hypothesis for the state.

A straight-forward implementation of SLAM using particle filters would scale exponentially with the number of landmarks in the map [108]. However, FastSLAM exploits the nature of SLAM with known correspondences that makes two disjoint map landmarks conditionally independent conditioned on the robot path and history of measurements. Thus, the full posterior $p(\boldsymbol{x}_{0:k}, \boldsymbol{m} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k})$ can be written as the following product:

$$p(\boldsymbol{x}_{0:k}, \boldsymbol{m} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k}) = \left[\prod_{i=0}^{N_m - 1} p(\boldsymbol{m}^i | \boldsymbol{x}_{0:k}, \boldsymbol{z}_{1:k}) \right] \cdot p(\boldsymbol{x}_{0:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k}). \tag{2.42}$$

The problem is decomposed into $N_m + 1$ estimation problems: N_m mapping problems (one for each landmark in the map) and one localization problem.

Rao-Blackwellized particle filters (see Section 2.2.6) are suitable filters to tackle this problem. Following (2.38) each particle represents a sample from $p(\boldsymbol{x}_{0:k} | \boldsymbol{z}_{1:k}, \boldsymbol{u}_{0:k})$, indicated by $\boldsymbol{x}_{0:k}^{[m]}$, and also up to N_m parametric representations of the map, $p(\boldsymbol{m}^i | \boldsymbol{x}_{0:k}^{[m]}, \boldsymbol{z}_{1:k})$ with $0 \leq i \leq N_m - 1$.

FastSLAM estimates the location of each landmark using one separate EKF [108]. This is the key difference with respect to EKF-SLAM, where there is a single EKF in charge of estimating the location of all landmarks in a joint manner.

Proposal Density Function

There are two main versions of the FastSLAM algorithm, which differ in the proposal density function used to sample new poses [108]:

- **FastSLAM 1.0** uses the control inputs in the proposal function:

$$q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}^{[m]}, \boldsymbol{u}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}^{[m]}, \boldsymbol{u}_{0:k}). \tag{2.43}$$

- **FastSLAM 2.0** draws particles from the following proposal function:

$$q(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}^{[m]}, \boldsymbol{u}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{1:k-1}^{[m]}, \boldsymbol{u}_{0:k}, \boldsymbol{z}_{1:k}), \tag{2.44}$$

  which also takes into account the measurements and is optimal in the Bayesian sense (Section 2.2.5).

FastSLAM 1.0 causes most particles to receive low weights during the weight update since the uncertainty regarding the control inputs is much higher relative to the uncertainty of the robot's sensors. Many particles with low weight do not survive resampling, making the algorithm inefficient. By taking into account the measurement, FastSLAM 2.0 makes the algorithm more efficient [155].

Computational Complexity of FastSLAM

It has been shown that a naïve implementation of the FastSLAM algorithm requires time in $\mathcal{O}(N_p N_m)$. The bottleneck is the resampling step where N_p' particles are drawn from the existing N_p particles and for each particle, its N_m map estimates are copied with replacement [108].

A more efficient implementation uses a tree-based data structure to store the maps [108]. Simply put, in such an implementation particles share parts of the map that are identical and results in computational complexity that grows with $\mathcal{O}(N_p \log N_m)$.

In Chapter 5, we will analyze the complexity of both a naïve implementation and a tree-based implementation of FastSLAM within the context of FootSLAM.

2.4 Human Odometry

In the robotics community, odometry refers to measurements of the revolutions of a robot's wheels that can be used to estimate changes in the state of the robot. Similarly, human odometry refers to the measurements of a pedestrian's steps that can be processed to track changes in his pose — his 3D position and orientation — over time.

2.4.1 Gait Cycle

The gait cycle can be divided into two phases: swing phase and stance phase. The stance phase starts when the pedestrian's heel touches the ground and finishes when the pedestrian lifts the foot from the ground. The swing phase encompasses the remaining

time when the foot is in the air until a new stance phase begins. The stance phase takes about 60% of the gait cycle, whereas the swing phase takes up the other 40% [161].

In this thesis the term "detecting a step" means detecting the stance phase of the gait cycle. A step measurement is defined as the distance and attitude change at the start of the stance phase since the last stance phase.

2.4.2 Collecting Step Measurements With Inertial Sensors

The most widely used sensors to measure a pedestrian's steps and track his pose without a deployed infrastructure are accelerometers, gyroscopes and magnetometers [71]. An accelerometer measures linear acceleration, a gyroscope measures angular velocity and a magnetometer measures the magnetic field vector. Integrating the readings of the triaxial gyroscope yields an estimate of the attitude change, while integrating twice the readings of the triaxial accelerometer yields an estimate of the change in position in 3D.

These sensors are found in many forms: built in Micro Electromechanical Systems (MEMS)-based IMUs but also in modern smartphones, tablets, smart watches [7] or shoes [113][10].

To this date, the most reliable approach to pedestrian tracking using inertial sensors is using a foot-mounted IMU [52][166][97][79][80][113][63]. In practice, travelers might use lightweight wireless clip-on sensors for their shoes — as it has become popular among sport enthusiasts [10] — or such sensors might even be integrated into shoes themselves [113]. This might be appropriate for security applications or early technology adopters, but mounting a sensor on the foot is often impractical for mass-market applications.

As a result, and given that the built-in sensors of smartphones and other mobile devices have become more accurate, the navigation community is also exploring the possibility of tracking a pedestrian's pose with other sensor placements: in-pocket [149][110], attached to the wrist [120] or allowing free motion, e.g. hand-held [1][78][127].

If the pedestrian can attach a sensor to the shoe, then given the proximity of the sensor to the foot, one can integrate the acceleration readings to compute the change in position of the pedestrian. This is known as strap-down Inertial Navigation System (INS).

2.4.3 Strap-down Inertial Navigation

Sensor measurements (specially those of low-cost sensors) are inaccurate and thus the estimates of pose change suffer from drifting errors that grow without bounds over time [165]. However, with foot-mounted sensors we can reliably detect periods of zero velocity during the stance phase.

In 2005 Foxlin introduced Zero Velocity Updates (ZUPTs) as a pseudo-measurement in an EKF [52] (Section 2.2.2). ZUPT pseudo-measurements allow the EKF to correct

position, velocity, accelerometer biases, pitch, roll, and the pitch and roll gyro biases after each stride. This reduces the error growth of inertial navigation from cubic to linear. Yaw (heading) and the yaw gyro bias are the only important EKF states that remain unobservable from ZUPT pseudo-measurements.

The magnetic field measurements can be used to correct short-term gyroscope biases, for example by using Magnetic Angular Rate Update (MARU) [166] or Quasi-static periods [1], which help stabilize the heading drift. When the sensor is mounted on the foot, we can also use Zero Angular Rate Update (ZARU) as means of heading drift stabilization [123].

2.4.4 Other Sensor Placements

Estimating the attitude and velocity of a pedestrian with sensor placements far from the foot remains still a challenging problem and constitutes a research topic in itself. These are the main challenges:

1. Periods of zero velocity cannot be detected accurately, and thus approaches such as ZUPT cannot be used. As a result, it is much harder to estimate and correct position, velocity, accelerometer biases, pitch, roll, and the pitch and roll gyro biases.
2. The angular displacement between the sensor frame and the body frame is usually unknown and can change over time, for example in the case of a hand-held smartphone or when carrying a smartphone in a backpack.

These challenges make it infeasible to directly integrate acceleration measurements to estimate the pedestrian's steps and step length. In this case typically the problem is separated into:

a) **Step detection** (also called step counting) [3][24][109]: A large number of algorithms, often based on biomechanical models of human motion, have been proposed. These algorithms include time domain, frequency domain and feature clustering approaches [24].

b) **Step length estimation**: A number of empirical formulas to estimate step length have been proposed [78][109]. Munoz et al. [109] propose a novel approach based on the opening angle of the leg to measure step length when the sensor is placed in the pocket or on the hip. Renaudin et al. use a linear relationship between step length and step frequency, weighted by the user's height (which needs to be also estimated) [127][126].

c) **Motion mode recognition**, for example to differentiate between texting, placing a phone call or carrying the phone in a bag [151] or to extract activities such as taking an elevator or climbing stairs [53].

d) **Attitude estimation** [104][1][110]: if the sensor is attached to the pedestrian's body and the relative orientation between the user's body frame and the sensor frame is

known, then we can estimate the pedestrian's heading.

e) **Elevation or floor number estimation**: This usually involves other sensors such as a barometer [48][101], which measures air pressure and under stable weather conditions, it can be used to measure elevation changes.

2.4.5 Approach Used in this Thesis

All the odometry datasets processed in this thesis have been collected with one single foot-mounted IMU and the sensor measurements have been processed using the UKF in [166]. This UKF substituted the EKF in the cascaded estimation architecture of [97] to better capture non-linearities.

Based on ZUPT measurements for stance detection, this implementation of the UKF uses MARUs [166] during the stance phase to estimate the gyro bias. The filter is composed of 15 states, with 3 states for each one of the following components: orientation (roll, pitch, yaw), position vector, velocity vector, accelerometer bias and gyroscope bias. With one single IMU we obtain one step measurement (step length and heading change) every two steps, at roughly 1 (double) step per second (Fig. 2.5).

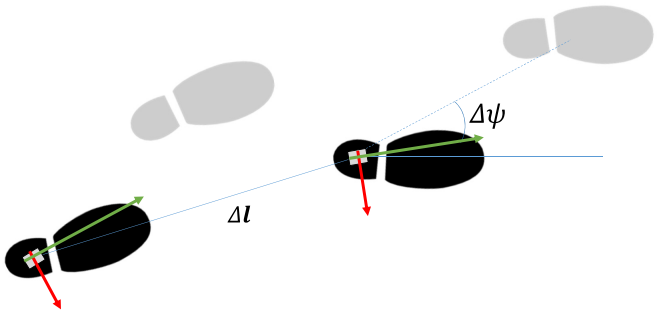


Figure 2.5: The step length Δl and heading change $\Delta\psi$ of a pedestrian are measured with a foot-mounted sensor. With one single sensor, we measure the pose change between every second step.

Figure 2.6 shows an example of odometry measurements collected by a pedestrian while walking with a foot-mounted IMU processed using the UKF of [166]. The measurements clearly suffer from heading drift, visible on the upper part of the figure where the two triangle-shaped paths should overlap with each other.

Note that the only requirement for FootSLAM is a source of stable odometry and other sensor placements are not precluded. For example, PocketSLAM [86] performs FootSLAM based on odometry collected with an IMU placed in the pocket.

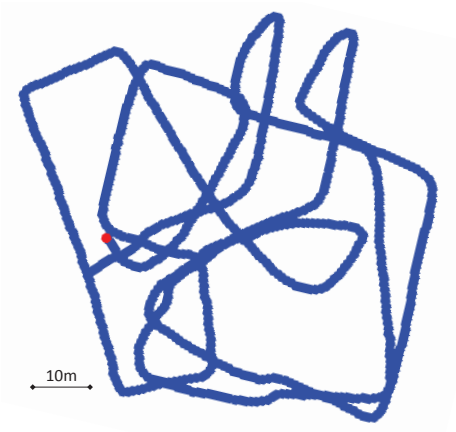


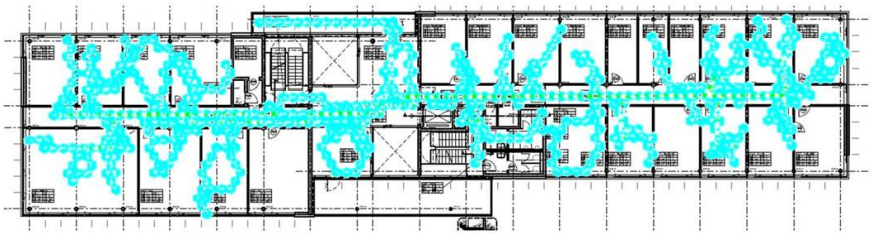
Figure 2.6: Example of odometry measurements collected by means of a foot-mounted IMU and processed using the UKF in [166]. The resulting odometry clearly suffers from heading drift.

2.5 The FootSLAM Principle

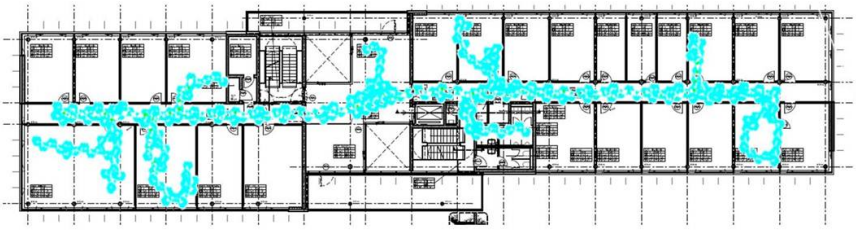
FootSLAM [8] addresses SLAM for pedestrians by jointly estimating the pose of the pedestrian — his position and heading — and the map of the environment following a Bayesian formulation, as new measurements become available. The input to FootSLAM are human odometry measurements and the resulting map consists of the step direction probabilities at each location that the pedestrian visits. It does not matter whether walls or furniture constrain human motion, all these aspects are reflected in the FootSLAM map.

FootSLAM processes human odometry measurements by means of a RBPF (Section 2.2.6) to estimate the drifting errors of the odometry and to learn the step direction probabilities at each location the pedestrian visits. To do that, the space is partitioned and each next step of the pedestrian is drawn from a location dependent probability distribution that only depends on his current position. FootSLAM rewards particles whose trajectory is more compatible with the learned map. When a pedestrian revisits similar transitions in the space, "closing the loop", particles that register previously observed spatial transitions are rewarded with a higher weight update. The result is a probabilistic map of human motion that can be used for localization.

Work on FootSLAM before this thesis has focused on generating accurate human motion maps in 2D environments [128][8]. Figure 2.7 depicts two step transition maps learned by the particle with the highest likelihood corresponding to two walks within an office building. On the image above, an extensive walk is depicted and the map encompasses almost all offices and other walkable areas. Below, the corresponding map of a shorter and less extensive walk is shown. Figure 2.8 shows the transition map of a building in Vienna (Austria) that presents a rather interesting layout, as learned by the particle with the highest likelihood.



(a) FootSLAM map resulting from an odometry dataset collected during a long walk.



(b) FootSLAM map resulting from an odometry dataset collected during a shorter walk.

Figure 2.7: Examples of FootSLAM maps resulting from human odometry collected at an office environment. The building plan is shown only as a reference.

From this section and forward we follow the notation first introduced in [128], whereby bold capitalized letters represent random variables.

2.5.1 Bayesian Formulation

In FootSLAM, given the history of (noisy) **step measurements** (collected as explained in Section 2.4), $\boldsymbol{Z}^{U}_{1:k}$, the goal is to jointly estimate the history of poses of the pedestrian $\boldsymbol{P}_{0:k}$ and the map of the environment $\boldsymbol{M}$ [128]. Note that in contrast to robotic SLAM, where the robot is controlled by a series of inputs that guide its movement, in pedestrian SLAM we do not have control over the pedestrian's path. Instead the pedestrian follows his intent **Int** guided by visual cues **Vis** to avoid walls and others obstacles. The state in FootSLAM is specified by the following unobservable variables (see DBN in Fig. 2.9):

1. The history of **poses** of the pedestrian, $\boldsymbol{P}_{0:k}$.
2. The history of **true step vectors** $\boldsymbol{U}_{0:k}$,
3. The history of correlated **error states**, $\boldsymbol{E}_{0:k}$ that distort the step measurements $\boldsymbol{Z}^{U}_{1:k}$.
4. The **time invariant map** of the environment in which the pedestrian walks, $\boldsymbol{M}$.

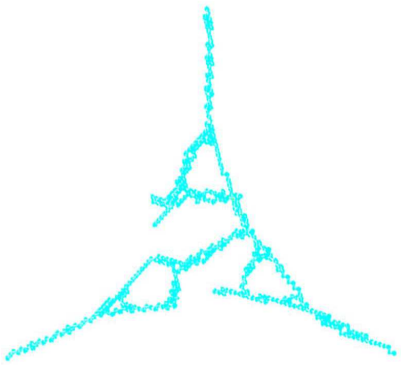


Figure 2.8: FootSLAM map resulting from an odometry dataset collected in a building with an unusual layout.

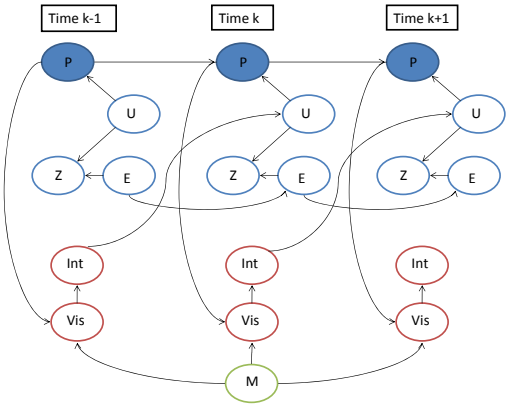


Figure 2.9: DBN of the FootSLAM estimation problem during three time slices. Source: [54].

Then, following a recursive Bayesian formulation [128], FootSLAM addresses the problem of estimating the full posterior:

$$p(\boldsymbol{P}_{0:k}, \boldsymbol{U}_{0:k}, \boldsymbol{E}_{0:k}, \boldsymbol{M}|\boldsymbol{Z}^{U}_{1:k}) = p(\{\boldsymbol{PUE}\}_{0:k}, \boldsymbol{M}|\boldsymbol{Z}^{U}_{1:k}). \tag{2.45}$$

Following the FastSLAM factorization [108], FootSLAM employs a RBPF to decompose the estimation problem into a pedestrian localization problem and a mapping problem conditioned on the pedestrian's poses. Mathematically, first we apply Bayes to the full posterior:

$$\begin{aligned} & p(\{\boldsymbol{PUE}\}_{0:k}, \boldsymbol{M}|\boldsymbol{Z}^{U}_{1:k}) \\ \overset{Bayes}{=} & \; p(\boldsymbol{M}|\{\boldsymbol{PUE}\}_{0:k}, \boldsymbol{Z}^{U}_{1:k}) \cdot p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^{U}_{1:k}), \end{aligned} \tag{2.46}$$

Given the history of poses of the pedestrian $\boldsymbol{P}_{0:k}$, the map $\boldsymbol{M}$ becomes conditionally independent of the history of step vectors $\boldsymbol{U}_{0:k}$, measurements $\boldsymbol{Z}^U_{1:k}$ and errors $\boldsymbol{E}_{0:k}$ (Fig. 2.9). As a consequence, the full posterior can be simplified:

$$p(\boldsymbol{M}|\{\boldsymbol{PUE}\}_{0:k}, \boldsymbol{Z}^U_{1:k}) \cdot p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^U_{1:k}) = \underbrace{p(\boldsymbol{M}|\boldsymbol{P}_{0:k})}_{\substack{\text{mapping problem} \\ \text{conditioned on the} \\ \text{pedestrian's poses}}} \cdot \underbrace{p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^U_{1:k})}_{\text{localization problem}}. \quad (2.47)$$

This equation is an example of the factorization of (2.38) (also known as Rao-Blackwellization). Each particle m in the RBPF represents:

$$\{p(\boldsymbol{M}|\boldsymbol{P}^{[m]}_{0:k}), p(\{\boldsymbol{PUE}\}^{[m]}_{0:k}|\boldsymbol{Z}^U_{1:k})\},$$

and as it will be shown shortly, a parametric representation is used for $p(\boldsymbol{M}|\boldsymbol{P}^{[m]}_{0:k})$.

FootSLAM represents human motion as a first order Markov process: The next step of the pedestrian depends only on his current location (see Fig. 2.9). In order to estimate the probability distributions of human motion as a function of location, FootSLAM partitions the 2D space into a grid of N_H adjacent regular hexagons $H_h \in \{H_0, H_1, \ldots, H_h, \ldots, H_{N_H-1}\}$, where the index h uniquely references a hexagon's position. The hexagon grid is composed of N_x columns and N_y rows, with $N_H = N_x \cdot N_y$ (Fig. 2.10(a)). Each hexagon H_h has 6 edges, which we denote with $e \in \{0, \ldots, 5\}$ (Fig. 2.10(b)) or with $e_{hj}\ \forall j$ s.t. $h \mapsto j$, where the symbol $\mapsto$ indicates that one can reach hexagon H_j from hexagon H_h crossing edge e_{hj} (Fig. 2.10(c)).

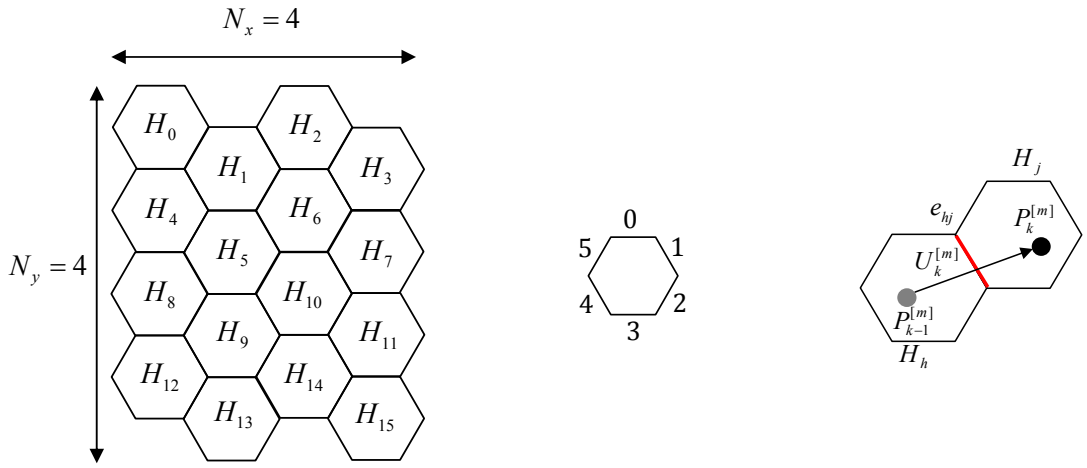
The indexing chosen for the grid of hexagons (Fig. 2.10(a)) allows us to quickly identify the hexagon at which a particle is located, which in turn is needed to quickly determine what edges the particle crossed when moving from one hexagon to the next (more insights can be found in Sections 5.2.6 and 5.2.7 in [22]).

2.5.2 Why Use a Dense Grid of Hexagons?

We characterize the influence of location on human motion as a local probability distribution of the direction of motion. In many indoor environments this distribution will tend to become independent after roughly one meter. Smaller displacements are very likely to share similar distributions.

One can tile the 2D space using triangles, rectangles or hexagons. Using such polygons leads to a simple mapping and weighting computation. These polygons allow either 90 degree or 60 degree discretization of direction of motion. We believe that the probabilities of the directions of a person's next step are approximately independent for discrete angular ranges of about 60 degrees. For these reasons and for simplicity we have chosen hexagons.

This thesis builds on prior work of FootSLAM based on a hexagonal grid [128][8][131]. There have been no strong reasons to deviate from the hexagonal discretization of the space. Other implementations are, of course, not precluded.



(a) Grid of hexagons used to discretize the 2D space, composed of $N_x = 4$ columns and $N_y = 4$ rows.

(b) Indices assigned to the edges of each hexagon in the grid.

(c) Transition from hexagon H_h to hexagon H_j crossing edge e_{hj}, corresponding to edge number 1 of hexagon H_h and edge number 4 of hexagon H_j.

Figure 2.10: Definition of indices and variables used for the discretization of the space in 2D.

2.5.3 Mapping Problem

FootSLAM maps are location-based maps with the map defined as the set:

$$M = \{M_0, M_1, \ldots, M_h, \ldots, M_{N_{H-1}}\}, \tag{2.48}$$

where M_h represents itself a set of transition probabilities $\{M_h^0, M_h^1, \ldots, M_h^5\}$ across the 6 edges of hexagon H_h. Unlike occupancy grid maps, whereby each cell is associated to a binary state representing whether it is occupied or not, FootSLAM maps provide a richer representation of each cell, indicating the probabilities of exiting a hexagon across each one of its edges:

$$M_{h(\boldsymbol{P}_{k-1})}^{e(\boldsymbol{U}_k)} = P(\boldsymbol{P}_k \in H_j | \boldsymbol{P}_{k-1} \in H_h), \tag{2.49}$$

and such that $H_h \neq H_j$. Therefore, it represents the probability of crossing edge e_{hj} when leaving hexagon H_h to go into hexagon H_j by adding step vector $\boldsymbol{U}_k$ to $\boldsymbol{P}_{k-1}$ (Fig. 2.10(c)). Also, we define M_h such that $\sum_{e=0}^{e=5} M_h^e = 1$.

Since we do not know M_h^e we let $\boldsymbol{M}_h^e$ denote the map random variable whose conditional distribution $p(\boldsymbol{M}_h^e | \boldsymbol{P}_{0:k})$ we can estimate as a result of the history of pedestrian poses $\boldsymbol{P}_{0:k}$ up to step k.

The assumption of the division of the space into independent hexagons makes possible the decomposition of the mapping problem into map estimation subproblems [128], one for each hexagon:

$$p(\boldsymbol{M}|\boldsymbol{P}_{0:k}) = \prod_{h=0}^{N_H-1} p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k}). \tag{2.50}$$

Finally, the full posterior of (2.47) can be written as:

$$p(\{\boldsymbol{PUE}\}_{0:k}, \boldsymbol{M}|\boldsymbol{Z}^U_{1:k}) = \underbrace{\prod_{h=0}^{N_H-1} p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k})}_{N_H \text{ map estimators}} \cdot \underbrace{p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^U_{1:k})}_{\text{one pose estimator}}, \tag{2.51}$$

Therefore, the full posterior can be divided into $(N_H + 1)$ estimators: N_H estimators for the computation of the posterior over the map and one for the computation over the pedestrian's poses, step vectors and correlated errors.

2.5.4 Localization Problem

The second term in (2.51) can be expressed recursively as:

$$p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^U_{1:k}) \propto p(\{\boldsymbol{PUE}\}_k|\{\boldsymbol{PUE}\}_{0:k-1}, \boldsymbol{Z}^U_{1:k}) \cdot \underbrace{p(\{\boldsymbol{PUE}\}_{0:k-1}|\boldsymbol{Z}^U_{1:k-1})}_{\text{recursion}}, \tag{2.52}$$

which yields, after using the conditional independence relationships encoded in the DBN in Fig. 2.9 [128]:

$$p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^U_{1:k}) \propto \overbrace{p(\boldsymbol{E}_k|\boldsymbol{E}_{k-1})}^{\text{error state transition}} \cdot \overbrace{p(\boldsymbol{Z}^U_k|\boldsymbol{U}_k\boldsymbol{E}_k)}^{\text{measurement likelihood}} \cdot \overbrace{p(\{\boldsymbol{PU}\}_k|\{\boldsymbol{PU}\}_{0:k-1})}^{\text{conditional pose transition probability}} \cdot \underbrace{p(\{\boldsymbol{PUE}\}_{0:k-1}|\boldsymbol{Z}^U_{1:k-1})}_{\text{recursion}}, \tag{2.53}$$

starting from an arbitrary pose $\boldsymbol{P}_0 = (x, y, z, \rho)$, where (x, y, z) is the pedestrian position in 3D (for now we consider $z = 0, \forall k$) and ρ the pedestrian's heading.

In the following we write h for the index of the outgoing hexagon $h(\boldsymbol{P}_{k-1})$ and e for the crossed edge $e(\boldsymbol{U}_k)$ for ease of notation (Fig. 2.10(c)). The third factor in (2.53), which we define as I, is the FootSLAM term [128]. It can be computed by marginalizing over

the map $\boldsymbol{M}$:

$$\begin{aligned} I \triangleq & \quad p(\{\boldsymbol{PU}\}_k|\{\boldsymbol{PU}\}_{0:k-1}) = \int_{\boldsymbol{M}} p(\{\boldsymbol{PU}\}_k|\boldsymbol{P}_{k-1},\boldsymbol{M}) \cdot p(\boldsymbol{M}|\boldsymbol{P}_{0:k-1})\ \mathrm{d}\boldsymbol{M} \\ = & \quad \int_{\boldsymbol{M}_0}\int_{\boldsymbol{M}_h}\int_{\boldsymbol{M}_j}\dots\int_{\boldsymbol{M}_{N_H-1}} p(\{\boldsymbol{PU}\}_k|\boldsymbol{P}_{k-1},\boldsymbol{M}_h) \cdot \prod_i p(\boldsymbol{M}_i|\boldsymbol{P}_{0:k-1})\ \mathrm{d}\boldsymbol{M} \\ = & \quad \prod_{i\backslash h}\Big[\int_{\boldsymbol{M}_i} p(\boldsymbol{M}_i|\boldsymbol{P}_{0:k-1})\ \mathrm{d}\boldsymbol{M}_i\Big] \cdot \int_{\boldsymbol{M}_h} p(\{\boldsymbol{PU}\}_k|\boldsymbol{P}_{k-1},\boldsymbol{M}_h) \cdot p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k-1})\ \mathrm{d}\boldsymbol{M}_h, \end{aligned} \tag{2.54}$$

where we have made use of the fact that the pose and true step $\{\boldsymbol{PU}\}_k$ are assumed to be conditionally independent of the history of poses previous to $k-1$ and steps previous to k conditioned on $\boldsymbol{P}_{k-1}$ and the map learned from those poses $\boldsymbol{M}$.

The integrals inside the product all equate to unity. Thus all that remains is the integral over $\boldsymbol{M}_h$:

$$I \triangleq \int_{\boldsymbol{M}_h} p(\{\boldsymbol{PU}\}_k|\boldsymbol{P}_{k-1},\boldsymbol{M}_h) \cdot p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k-1})\ \mathrm{d}\boldsymbol{M}_h. \tag{2.55}$$

To solve this equation we will assume that $p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k-1})$ follows a Dirichlet distribution.

Dirichlet Distribution as Conjugate Prior of Multinomial Distribution

Let's assume that each one of the hexagons in the map corresponds to an unfair dice. The same way the transition probabilities across each one of the edges is unknown to us, the probability of each side of the dice showing up when rolling the dice is unknown to us.

Although the probability of each side of the dice is unknown, one can roll the dice n times and count the number of times each one of the $E = 6$ sides shows up. These data we get from observation follows a multinomial distribution. Remember: the multinomial distribution is the multivariate generalization of the binomial distribution and gives the probability of a particular combination of counts for each side.

One could try to use the multinomial distribution to compute the probability of each side showing up. However, this will be a very poor estimate within the first few dice rolls. A much better approach is to incorporate prior expectations (in statistics just called prior). In Bayesian statistics this is done using a Dirichlet distribution.

The Dirichlet distribution is the multivariate generalization of the Beta distribution and it is the conjugate prior of the multinomial distribution. This means that if the data we get from observations follows a multinomial distribution and we assume a prior Dirichlet distribution, then the posterior distribution also follows an (updated) Dirichlet distribution [72].

Intuitively we would like to start with a uniform probability across the sides of the dice (a fair dice). Upon rolling the dice and observing what side shows up, we can update our knowledge and learn how likely it is for a side to show up.

In FootSLAM we assume that $p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k-1})$ follows a Dirichlet distribution. Conceptually, this Dirichlet distribution represents the belief that the probabilities of $E = 6$ mutually exclusive events (a hexagon's edge being crossed) are $\{M_h^0, \ldots, M_h^5\}$, i.e. $\{p(\boldsymbol{M}_h^0 = M_h^0|\boldsymbol{P}_{0:k-1}), \ldots, p(\boldsymbol{M}_h^5 = M_h^5|\boldsymbol{P}_{0:k-1})\}$ given that each event has been observed $u_h^e - 1$ times. Thus, for each hexagon H_h, the Dirichlet distribution is parametrized by a vector with six positive real numbers: $\{u_h^0, \ldots, u_h^5\}$ and with $u_h^T = \sum_{e=0}^{5} u_h^e$, which can be written as:

$$\mathcal{D}(p(\boldsymbol{M}_h^0 = M_h^0|\boldsymbol{P}_{0:k-1}), \ldots, p(\boldsymbol{M}_h^5 = M_h^5|\boldsymbol{P}_{0:k-1}); u_h^0, \ldots, u_h^5) = \frac{1}{\mathrm{B}(u_h^0, \ldots, u_h^5)} \prod_{e=0}^{e=5} M_h^e\,, \tag{2.56}$$

where:

$$\mathrm{B}(u_h^0, \ldots, u_h^5) = \frac{\prod_{e=0}^{e=5} \Gamma(u_h^e)}{\Gamma(u_h^T)}, \tag{2.57}$$

is the Beta function and:

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} \mathrm{d}t, \tag{2.58}$$

is the Gamma function.

The expected probability of crossing edge e of hexagon H_h then becomes [72]:

$$E[p(\boldsymbol{M}_h^e = M_h^e|\boldsymbol{P}_{0:k-1})] = \frac{u_h^e}{u_h^T} \tag{2.59}$$

In FootSLAM we choose a uniform prior probability across all edges of any hexagon: $\alpha_h^e = 0.8$ and $\alpha_h^T = \sum_{e=0}^{e=5} \alpha_h^e$. The term α_h^e refers to the prior virtual counts, chosen empirically to be 0.8 for all edges when no other information is available. Thus $E[p(\boldsymbol{M}_h^e = M_h^e)] = 1/6 \;\; \forall\, e$ in the absence of edge transitions.

In our PF implementation, to learn the local map $\boldsymbol{M}_h^{[m]}$, for each hexagon that a particle visits, we count the number of times the particle m has crossed that edge, C_h^e. To store the counts we use a data structure called Hexagon Transition Counters (HTC). An HTC is a vector composed of six elements. Each vector element registers the number of transitions C_h^e across an edge. The associated HTC to hexagon H_h that has been visited by particle m is:

$$\{C_h\}^{[m]} = \{C_h^0, C_h^1, C_h^2, C_h^3, C_h^4, C_h^5\}^{[m]}. \tag{2.60}$$

Note that when written in bold face, $\boldsymbol{M}_h$ denotes the unknown random variable that we wish to estimate: The transition probabilities across the edges of a hexagon. On the other hand, C_h represents an HTC, i.e. the vector of 6 hexagon transition counters from which the probabilities can be inferred [128].

Given an initial prior Dirichlet distribution:

$$\mathcal{D}(p(\boldsymbol{M}_h^0 = M_h^0), \ldots, p(\boldsymbol{M}_h^5 = M_h^5); \alpha_h^0, \ldots, \alpha_h^5),$$

and having crossed the edges of the hexagon $\{C_h^0, \ldots, C_h^5\}$ times as result of a pedestrian's pose sequence $\boldsymbol{P}_{0:k-1}$, the posterior distribution follows:

$$\mathcal{D}(p(\boldsymbol{M}_h^0 = M_h^0|\boldsymbol{P}_{0:k-1}), \ldots, p(\boldsymbol{M}_h^5 = M_h^5|\boldsymbol{P}_{0:k-1}); \alpha_h^0 + C_h^0, \ldots, \alpha_h^5 + C_h^5).$$

Then using the fact that $p(\{\boldsymbol{PU}\}_k^{[m]}|\boldsymbol{P}_{k-1}^{[m]}, \boldsymbol{M}_h) \propto \boldsymbol{M}_h^e$ (per (2.49)) and that $p(\boldsymbol{M}_h|\boldsymbol{P}_{0:k-1}^{[m]})$ follows a Dirichlet distribution with observed counts $\{C_h^0, \ldots, C_h^5\}$, Equation (2.55) yields:

$$I^{[m]} \propto \left\{ \frac{C_h^e + \alpha_h^e}{C_h^T + \alpha_h^T} \right\}^{[m]}, \tag{2.61}$$

where C_h^e are the number of transitions across the crossed edge e of hexagon H_h in the map of the particle m computed up to step k and $C_h^T = \sum_{e=0}^{e=5} C_h^e$.

If a particle crosses several edges with one single step measurement, this translates into a product of probabilities for all edges crossed [128].

2.5.5 Particle Map and Cumulative Map

Particle Map

As mentioned earlier, in FootSLAM at time stamp k, each particle m in the RBPF represents $\{p(\boldsymbol{M}|\boldsymbol{P}_{0:k}^{[m]}), \{\boldsymbol{PUE}\}_k^{[m]}\}$.

For the particles to effectively represent $p(\boldsymbol{M}|\boldsymbol{P}_{0:k}^{[m]})$, each particle is assigned its own data structure to store the associated map estimates corresponding to the pedestrian's history of poses $\boldsymbol{P}_{0:k}^{[m]}$. This structure is referred to as *particle's map* or, in short, map.

Each particle's map, $\{M\}^{[m]}$, comprises the set of visited hexagons $\mathcal{N}_{\text{vis}}^{[m]}$ with their corresponding HTCs, i.e.:

$$\{M\}^{[m]} = \{H_h, \{C_h\}^{[m]}\}; \forall H_h \in \mathcal{N}_{\text{vis}}^{[m]}. \tag{2.62}$$

The work previous to this thesis builds on a naïve FootSLAM implementation, which stores each particle's map using a lookup table. Each entry in the table corresponds to a visited hexagon and its HTC. This implementation is memory and computationally inefficient because:

1. Particles that share the same history of poses (for example, after resampling) will have duplicated entries in their corresponding lookup tables.

2. In the resampling step, N_p' new particles are drawn from the existing set of N_p particles according to their importance weight $w_k^{[m]}$ (see Section 2.2.5). In FootSLAM, the entire map $\{M\}^{[m]} = \{H_h, \{C_h\}^{[m]}\}$ needs to be copied (with replacement) from the resampling particle. But the size of the lookup table will grow as the pedestrian explores new areas, making resampling more and more computationally inefficient over time. In particular, this implementation exhibits a computational complexity that grows with $\mathcal{O}(t^2)$, preventing the FootSLAM algorithm from being executed in real-time.

In Chapter 5, we will analyze the complexity of the current algorithm in more detail and propose a reduced-complexity implementation, which will allow real-time mapping of larger areas.

Cumulative Map

The hexagon transition estimates of the N_p particles' maps can be compressed in the form of a *cumulative map*, also called *expected map* in [21]. This cumulative map $\{M\}^{\text{cum}}$ contains the hexagons visited by *all* particles with their corresponding cumulative HTCs, $\{C_h\}^{\text{cum}}$:

$$\{M\}^{\text{cum}} = \{H_h, \{C_h\}^{\text{cum}}\};\ \forall H_h \in \mathcal{N}_{\text{vis}}^{\text{cum}}, \tag{2.63}$$

where $\mathcal{N}_{\text{vis}}^{\text{cum}} = \cup\, \mathcal{N}_{\text{vis}}^{[m]}$; for $m = 0, \ldots, N_p - 1$.

The cumulative HTC is computed using the weights of each particle $w^{[m]}$:

$$\{C_h\}^{\text{cum}} = \{C_h^0, C_h^1, C_h^2, C_h^3, C_h^4, C_h^5\}^{\text{cum}}, \tag{2.64}$$

where:

$$\{C_h^e\}^{\text{cum}} = \sum_{m=0}^{N_p-1} w^{[m]} \cdot \{C_h^e\}^{[m]}. \tag{2.65}$$

In this thesis, we also refer to these maps as *posterior* maps. Note, however, that this compression loses information contained in the full posterior.

2.5.6 Sequential Estimation

Importance Density

To allow for sequential estimation (2.30), the RBPF implementation of FootSLAM samples — using from hundreds to tens of thousands of particles — from the following recursive importance density function [8]:

$$q(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}_{1:k}^U) = q(\{\boldsymbol{PUE}\}_k|\{\boldsymbol{PUE}\}_{0:k-1}, \boldsymbol{Z}_{1:k}^U) \cdot q(\{\boldsymbol{PUE}\}_{0:k-1}|\boldsymbol{Z}_{1:k-1}^U).$$

When using odometry measurements provided by the lower-level UKF of [166], the full measurement likelihood $p(\boldsymbol{E}_k|\boldsymbol{E}_{k-1}) \cdot p(\boldsymbol{Z}_k^U|\boldsymbol{U}_k\boldsymbol{E}_k)$ has a much narrower variance compared to the pose transition probability I (2.55). Accordingly the posterior is closer in similarity to the measurement likelihood than to the pose transition probability. In such cases looking at the measurements $\boldsymbol{Z}_{1:k}^U$ improves performance (see Section 2.2.5).

In particular, the importance density has been chosen as [128]:

$$q(\{\boldsymbol{PUE}\}_k|\{\boldsymbol{PUE}\}_{0:k-1}, \boldsymbol{Z}_{1:k}^U) \triangleq p(\boldsymbol{E}_k|\boldsymbol{E}_{k-1}^{[m]}) \cdot p(\boldsymbol{U}_k|\boldsymbol{Z}_k^U\boldsymbol{E}_k^{[m]}). \tag{2.66}$$

Weight Update

The weight associated to each particle m is updated using the formula for a particle filter weight update (2.29), which for the importance density function in FootSLAM resolves to [8]:

$$w_k^{[m]} \propto \frac{p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}_{1:k}^U)}{q(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}_{1:k}^U)} \propto w_{k-1}^{[m]} \cdot \left\{\frac{C_h^e + \alpha_h^e}{C_h^T + \alpha_h^T}\right\}^{[m]}. \tag{2.67}$$

2.5.7 Error Models

The main error sources that distort 2D odometry measurements are explained in [128] and [134]. The following error components are considered (see also Algorithm 3 in [134]):

- Additive white translational noise.
- White noise on the estimated heading change.
- Additive colored angular error between the heading of the pedestrian and that measured since the last step. This is known as odometry heading drift and it is the most visible error source. The heading drift is encoded in the error state and follows a random walk process of order 1 with additive zero mean Gaussian noise with a standard deviation that is proportional to the square root of the time elapsed [134].

2.5.8 RBPF Algorithm in FootSLAM

A summary of the RBPF algorithm in FootSLAM is recalled here [8]:

1) Initialize all N_p particles pose $\boldsymbol{P}_0^{[m]} \leftarrow (x, y, z = 0, \rho = 0)$ and draw initial error states $\boldsymbol{E}_0^{[m]}$ from a proposed initial distribution.

2) For each time step k when a new step measurement is available:

   a) Draw particles from the proposal density in (2.66) from left to right: Sample from $p(\boldsymbol{E}_k|\boldsymbol{E}_{k-1}^{[m]})$ and add the drawn error vector to $\boldsymbol{Z}_k^U$ to compute $\boldsymbol{U}_k$.

b) For all particles, compute the new pose of the pedestrian $\mathbf{P}_k^{[m]}$ by adding the step vector $\boldsymbol{U}_k^{[m]}$ to $\boldsymbol{P}_{k-1}^{[m]}$.

c) Update the weight $w_k^{[m]}$ of each particle m using (2.67).

d) Normalize particle's weights to sum to unity.

e) Update the transition counters of the crossed edges corresponding to step starting in $\boldsymbol{P}_{k-1}^{[m]} \in H_h$ and finishing in $\boldsymbol{P}_k^{[m]} \in H_j$. In FootSLAM we assume that motion is equally likely backwards and forward and update the counters of both the outgoing hexagon and the incoming hexagon: $C_h^{e_{hj}}$ and $C_j^{e_{jh}}$, respectively. See more details below.

f) Resample if required.

In a nutshell, particles that revisit similar transitions are rewarded thus allowing the generation of a reliable map of the walked areas when the pedestrian closes a loop.

Update of Hexagon Transition Counters (Mapping)

The map update consists of the addition of one count to the counters of those edges that were crossed when particle m moved from $\boldsymbol{P}_{k-1}^{[m]}$ to $\boldsymbol{P}_k^{[m]}$.

For example, if particle m crossed edge $e = 2$ of hexagon H_h, such a transition is mapped as follows:

$$\{C_{h\rightarrow e=2}\}^{[m]} = \{C_h^0, C_h^1, \mathbf{C_h^2 + 1}, C_h^3, C_h^4, C_h^5\}^{[m]}, \tag{2.68}$$

where $C_{h\rightarrow e=2}$ indicates that the HTC has been updated after crossing edge 2.

The update of the HTC is the most fundamental step in the RBPF because it represents the basis for the particle weight computation (step 2c). If particle m revisits hexagon H_h crossing an edge that it had already crossed, it will be rewarded with a greater weight update than the other particles that are not revisiting edge transitions. As a result of this reward, the particle's hypotheses for the history of pedestrian's poses, step vectors and errors $\{\boldsymbol{PUE}\}_{0:k}^{[m]}$ will be considered to be more likely the one that the pedestrian and the system actually followed.

2.5.9 Open Research Questions

The accuracy of FootSLAM maps is governed by the average dimension of the physical structures (i.e. 1 m around doors, 2 m along corridors). Revisiting areas helps the estimator converge more quickly, but it is limited to 2D exploration. In addition, processing the collected data by a walking pedestrian is performed in an offline fashion, due to memory and computational constraints.

This thesis addresses the following topics to provide more insights into the FootSLAM approach and improve its performance:

1. Interpretation of a pedestrian's sequence of steps on a FootSLAM map as a random walk on a weighted graph (Chapter 3).

2. Calculation of the uncertainty of FootSLAM probabilistic maps as an indicator of the quality of the resulting map (Chapter 3).

3. Extension of FootSLAM to allow mapping multistory environments (Chapter 4).

4. Analysis of the computational complexity of FootSLAM, identification of main bottlenecks and reduction of its complexity for real-time localization and mapping of large areas (Chapter 5).

2.6 Crowdsourcing FootSLAM: FeetSLAM

In *crowdsourcing* [75] a group of people or community performs a task that had been traditionally assigned to specific individuals. Building on FootSLAM, FeetSLAM [136] maps indoor environments in a crowdsourced manner.

In FeetSLAM, a number of pedestrians N_d walk through accessible rooms and areas of a building collecting measurements of their steps actively or passively. We can envision three different scenarios [54]:

1. All walks start with the same initial pose or finish with the same end pose (or both) and overlap in explored area to a certain degree.

2. The walks do not necessarily start or finish with the same pose but overlap in explored area to a certain degree.

3. The walks do not necessarily start or finish with the same pose and do not necessarily overlap in the explored area.

For example, we might have a number of datasets obtained from an office environment whose corresponding FootSLAM maps may be inaccurate because FootSLAM did not reach convergence, e.g. if the pedestrian did not revisit areas ("closing the loop"). The key idea behind FeetSLAM is to use the observed spatial transitions of other pedestrians in the FootSLAM estimation process of a dataset collected by a different pedestrian. After jointly processing the datasets, they reach convergence and their combined map represents all visited areas (Figure 2.11). Areas in which the pedestrians were walking more often (darker hexagons) help reach convergence and can be represented more accurately.

The challenge of FeetSLAM is finding a suitable estimator that scales well with the number of collaborating pedestrians and the area explored by those pedestrians.

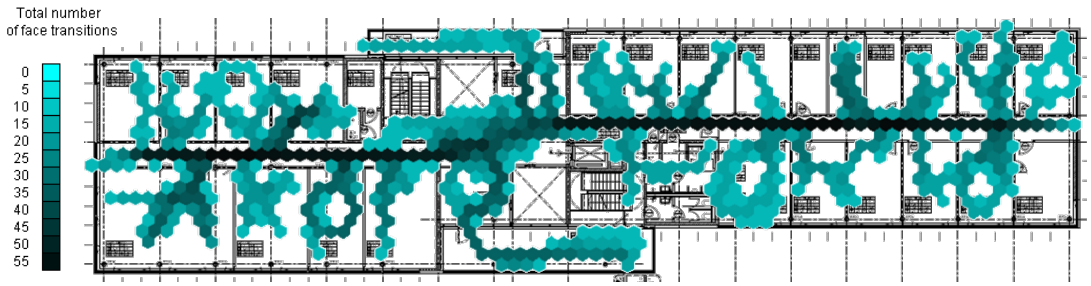


Figure 2.11: Combined FootSLAM map obtained from six walks in an office environment (in aquamarine) after 10 iterations using the FeetSLAM algorithm. The total duration of these 6 datasets was roughly 66 minutes. Darker hexagons represent hexagons with more edge transition counts that correspond to more frequented areas. The building layout was not used during FeetSLAM, but it is shown as a reference.

2.6.1 The Optimal Estimator

The extension of FootSLAM to the multiuser scenario processes all collected datasets (in parallel or sequentially) and needs to include the unknown starting conditions $\boldsymbol{SC}$ (initial pose parameters and their associated uncertainties) in the state [54]. The environment is the binding element between the pedestrians. The Bayesian estimator for a N_d-pedestrian scenario addresses the computation of the following posterior [136]:

$$p(\{\boldsymbol{PUE}\}_{0:k}^{1:N_d}, \boldsymbol{SC}^{1:N_d}, \boldsymbol{M} | \{\boldsymbol{Z}_{1:k}^{U}\}^{1:N_d}). \tag{2.69}$$

Figure 2.12 shows the DBN for a 2-pedestrian scenario. Note that for this DBN, the time indexes k are the same for the walks. This is not a requirement for the map merging algorithm, in which the data are processed offline, and hence can be obtained from walks occurring at different times. Conceptually, this means pairing the time indexes k of the walks and using padding (adding samples with value 0) when the walks are not equally long.

An example of this optimal estimator for collaborative SLAM can be found in the robotic domain in [74]. The approach builds on a single RBPF that estimates the map and the history of poses of the robots assuming that the relative pose between the robots can be accurately determined when the robots encounter each other.

Unfortunately, using a particle filter either in parallel or sequentially to process a large number of datasets, particles would suffer from particle depletion [137], since they would have to explore the state space of all odometry error sequences $\boldsymbol{E}_{0:k}^{1:N_d}$ for all the pedestrians and all starting conditions $\boldsymbol{SC}^{1:N_d}$. In addition, also due to depletion, when processing the datasets in a sequential fashion, FootSLAM would be biased towards the maps generated in the early stages.

But not everything is lost. In [136] and [54], it was shown that the DBN of Fig. 2.12

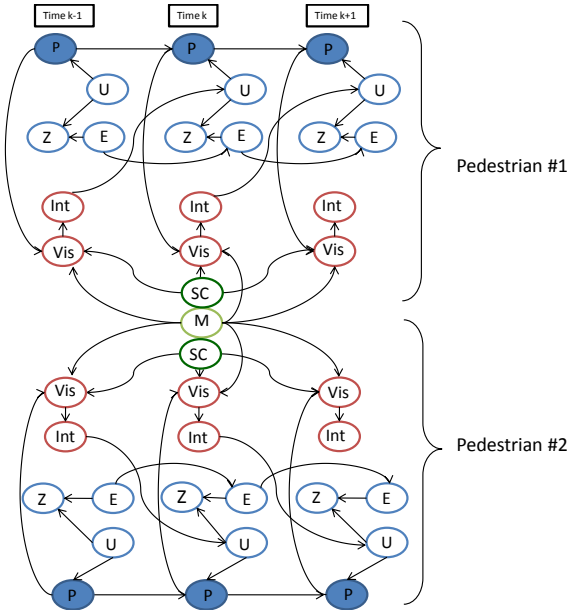


Figure 2.12: DBN for the FeetSLAM estimation problem with two pedestrians during three time slices (extracted from [54]).

shares some similarities with a family of codes, known as "Turbo"' codes [17], which can be decoded with a – suboptimal but less complex – iterative detector that performs with good error correction.

2.6.2 Suboptimal Estimator

The proceedings paper in [136] introduced "Turbo" FeetSLAM, a suboptimal estimator for multiuser mapping that borrows the following heuristics from "Turbo" codes [17]:

1. The problem is decomposed into smaller segments.

2. The data are processed in an iterative fashion.

3. Each stage feeds the next with "prior" information.

In a nutshell, given N_d datasets, at each iteration the iterative "Turbo" FeetSLAM algorithm processes independently each dataset using FootSLAM (step 1) and stores the resulting *posterior* FootSLAM maps. Then, it computes a *prior* map for the next iteration for each dataset (step 3) by combining the other $N_d - 1$ maps (see diagram in Fig. 2.13). Proceeding like this, the individual FootSLAM estimation runs are benefited from the knowledge of the other walks observed transitions, acting as a "loop closure" event.

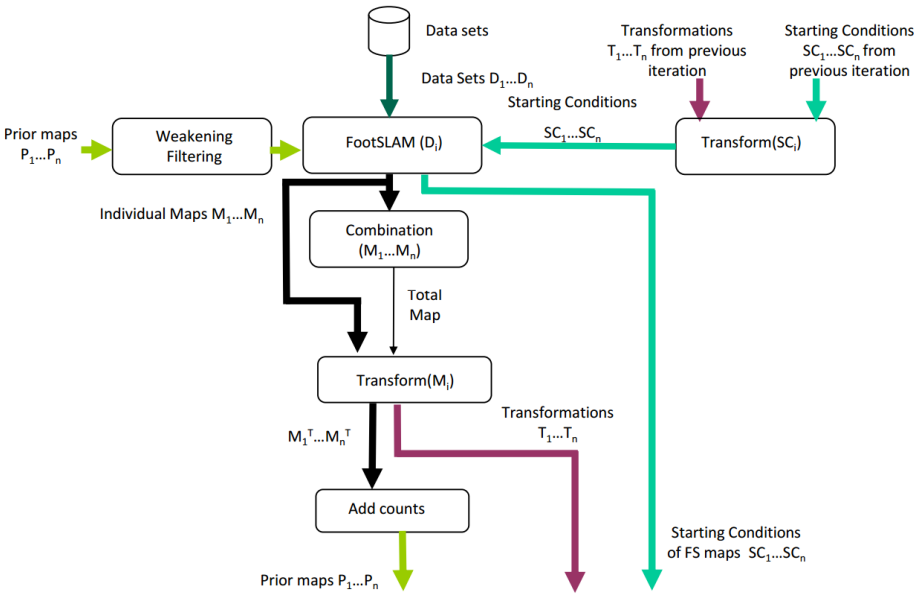


Figure 2.13: Diagram illustrating the "Turbo" FeetSLAM algorithm. Extracted from [136].

Using Other Maps in the FootSLAM Estimation Process

To incorporate prior knowledge about the transition counts, one can extend FootSLAM's weight update equation (2.67) to take into account the counts from another map:

$$w_k^{[m]} \propto w_{k-1}^{[m]} \cdot \left\{ \frac{C_h^e + \alpha_h^e + \beta_h^e}{C_h^T + \alpha_h^T + \beta_h^T} \right\}^{[m]}, \tag{2.70}$$

where β_h^e is the prior count for edge e of hexagon H_h and $\beta_h^T = \sum_{e'=0}^{5} \beta_h^{e'}$. This is the result of assuming that the prior distribution of each local hexagon's map follows a Dirichlet distribution with parameters:

$$\mathcal{D}(p(M_h^0 = M_h^0 | \boldsymbol{P}_{0:k-1}), \ldots, p(M_h^5 = M_h^5 | \boldsymbol{P}_{0:k-1}); \alpha_h^0 + \beta_h^0, \ldots, \alpha_h^5 + \beta_h^5).$$

Note that in order to use a map as a prior for a given dataset, the map shall not contain any contributions from that dataset. This is an important requirement that must be held in order to provide the dataset with information from other walks while avoiding biasing it with its own contribution. This is analogous to the prior construction in "Turbo" decoding and loopy belief propagation [105].

Controlling the Influence of the Prior Map

To avoid that a dataset dominates the resulting combined map and following concepts from simulated annealing [91], it was proposed to modify the prior maps by applying a

weakening function and a *smoothing* function to its transition counts [136][54]:

- The weakening function divides the transition counts of a map by a weakening factor that is greater than 1. The effect is controlling the influence of the prior map on the weight update of (2.70).

- The smoothing function spreads the transition counts of each edge among itself and that same edge of the six neighboring hexagons. The effect can be seen as a blurring function, allowing particles to be rewarded when crossing similar transitions in the vicinity.

Over the iterations these modifications are gradually reduced, allowing the improvements of each individual map to be propagated to other maps by means of the prior.

Proceeding like this, after each iteration the resulting individual and global FootSLAM maps become more accurate [136]. With this iterative processing the combined map is not dominated by one dataset, but it is the result of balancing the characteristics of each dataset with the effect of averaging out errors.

However, note that including other walks in a given FootSLAM estimation process or combining a number of maps is only possible, if we are able to relate all walks within the same coordinate system. In the absence of absolute anchors (e.g. GPS measurements) and given that FootSLAM maps are translation and rotation invariant, FootSLAM maps will not necessarily share the same coordinate system (Fig. 2.14). Thus, we need to find the geometric transformations that relate all the individual maps within the same coordinate system.

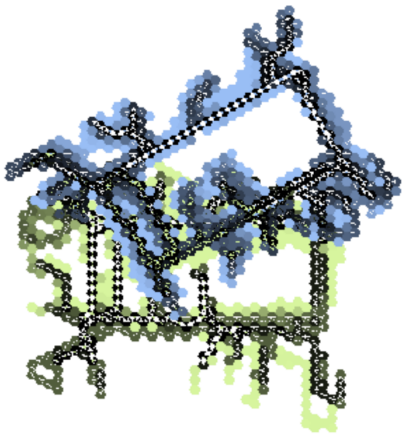


Figure 2.14: Two FootSLAM maps (depicted in blue and green) that are the result of processing two odometry datasets collected within the same building, but using different coordinate systems. Figure extracted from [136].

Geometric Transformation between Two Maps

Given two map instances of the same environment, M^P and M^Q (see example in Fig. 2.14), the existing implementation of the "Turbo" FeetSLAM algorithm [136] addresses the geometric transformation search problem by applying a **brute-force approach**: we apply different transformations – different sets of angular and spatial displacements – to one of the maps, we project the transition counts of both maps to a common grid of hexagons and rely on a log-likelihood-based correlation function to determine the best fit between them.

The projection onto a common grid of hexagons is necessary since after applying a geometric transformation to one of the maps, their hexagons will not necessarily be aligned. The common grid, called *target* grid in [136], can be that of the map that remains unchanged. In [136] the geometric transformation and the projection of the transition counts onto a target grid are performed in one step. Two factors indicate the proportion of the transition counts of each transformed edge that are assigned to the edges of the target grid (see details in Section 3.2 of [136] and in [54]):

1. A **distance factor** takes into account the relative distance between each transformed hexagon and the hexagons in the target grid.
2. An **angular factor** takes into account the relative orientation between the transformed edge and the target edges.

The log-likelihood based correlation function used to select the geometric transformation that makes one map M^P explain best the transitions of another map M^Q is defined as [136]:

$$\begin{aligned} \text{logLV}(M^P, M^Q) &= \frac{\sum_{h \in H_{M^Q}} \sum_{e=0}^{e=5} C_h^{eQ} \cdot \log\left(\frac{C_h^{eP} + \alpha_h^e}{C_h^{TP} + \alpha_h^T}\right)}{\sum_{h \in H_{M^Q}} C_h^{TQ}} \\ &+ \frac{\beta \cdot \sum_{h \in H_{M^Q}} C_h^{TQ} \cdot C_h^{TP}}{\sum_{h \in H_{M^Q}} C_h^{TQ}}, \end{aligned} \tag{2.71}$$

where β is a hexagon correlation factor, empirically chosen as $\beta = 0.04$, H_{M^Q} is the set of hexagons of map M^Q and the superindices P and Q indicate that the counts are those of M^P and M^Q, respectively.

In practice, to find the geometric transformation between two maps with manageable complexity, the transformation search space is limited to a finite range using the uncertainty regarding the starting conditions [136][54]. For now, we assume that the relative pose between the collaborating pedestrians is more or less well-known. In real-world scenarios,

the pedestrian might carry a GNSS receiver and have visibility to a sufficient number of satellites briefly during her walk, for example, before entering the building [131].

Computational Complexity of the Transformation Search

The map combination step in [136] applies N_x^{tr} translation values along the x axis times N_y^{tr} translation values along the y axis times N_r rotations values to M^Q and relies on the Bayesian-motivated function in (2.71) to determine the best fit to M^P.

The computational complexity of cross-correlating two maps using (2.71) grows in the number of hexagons of the maps $\mathcal{O}(N_H)$. This operation is performed $N_r \cdot N_x^{\text{tr}} \cdot N_y^{\text{tr}}$ times, which is of the order of $\mathcal{O}(N_x \cdot N_y) = \mathcal{O}(N_H)$ since we can consider N_r constant, i.e. $N_r = O(1)$. Thus, the total complexity order is $\mathcal{O}(N_H \cdot N_H) = \mathcal{O}(N_H^2)$: the "Turbo" FeetSLAM algorithm exhibits a computational complexity that grows quadratically in the area explored.

Computational Complexity of Combining N_d Maps

At each iteration, the algorithm needs to combine the N_d available individual maps into one single map (and eventually subtract the one map for which we are computing the prior map). To do this, in [136] each iteration is divided into $N_d - 1$ stages at which two maps are combined. To select which pair to combine next in an optimal manner, it was proposed to find the geometric transformation between *all* remaining combinations of maps, and then combine the pair with the highest log-likelihood value (2.71). This requires $(N_d - 1)^2$ map combinations at each iteration [136][54], which grows quadratically in the number of datasets, $\mathcal{O}(N_d^2)$ (more details can be found in Section 6.6 in [54]).

Although this procedure is accurate and robust, it does not scale well with the number of datasets and it is very time-inefficient.

2.6.3 Open Research Questions

The main challenge and bottleneck of the "Turbo" FeetSLAM algorithm is finding the geometric transformation that places all collected datasets within the same coordinate system.

This thesis addresses the following topics that improve and scale the performance of FeetSLAM:

1. Introduce an entropy-based metric to select the order in which the maps should be combined, reducing the dependency of the map combination step from quadratic to linear in the number of datasets (Chapter 3).

2. Derive an entropy-based indicator of the degree of similarity between two maps, which may serve as an alternative to (2.71) to select the transformation that makes one map explain best the transitions of another map (Chapter 3).
3. Propose an alternative approach to reduce the computational complexity of the brute-force transformation search approach of [136] (Chapter 6).
4. Extend the combination of the maps to multistory environments (Chapter 6).
5. Analyze the population requirements, computation time and memory needed to map the whole indoor world under certain assumptions and outline applications and deployment alternatives for crowdsourced mapping (Chapter 7).

3 FootSLAM Maps from an Entropy Perspective

Chapter 2 has given an introduction to Bayes filters, SLAM, human odometry and the pedestrian localization and mapping techniques FootSLAM and FeetSLAM. FootSLAM maps are a probabilistic representation of human motion. These maps are intrinsically uncertain because they are inferred from noisy step measurements, collected by a pedestrian with free will and because the pedestrian's spatial transitions are mapped using a discrete representation of the space (a grid of hexagons).

In order to better understand the uncertainty regarding human motion, in this chapter we introduce a new interpretation of a pedestrian's sequence of steps conditioned on a map using a random walk on a weighted graph. This interpretation provides a novel understanding of the inner workings of FootSLAM.

In addition, we present a number of entropy-based metrics to measure the uncertainty of FootSLAM maps and the uncertainty regarding a pedestrian's next step conditioned on a map. These metrics will be used to answer the following questions:

1. **How good is a FootSLAM map?** The uncertainty of the map and the uncertainty regarding the pedestrian's next step conditioned on that map can be used to measure the quality of the map and to evaluate the performance of the FootSLAM estimation algorithm.

2. **Can we find the geometric transformation between two maps using entropy metrics?** We will apply relative entropy metrics to find the geometric transformation that places two maps within the same coordinate system in order to combine them.

3. **Can we reduce the computational complexity of the "Turbo" FeetSLAM algorithm?** We will propose sorting map instances of the same environment by increasing uncertainty. Then the "Turbo" FeetSLAM algorithm shall add one map at a time to the existing combined map, starting with the maps with the lowest uncertainty. This will reduce the computational complexity of combining N_d maps from $\mathcal{O}(N_d^2)$ to $\mathcal{O}(N_d)$.

The main contributions of this chapter, authored by Kaiser et al. have been published in [84].

This chapter is organized as follows: Section 3.1 defines uncertainty in the context of FootSLAM probabilistic maps and reviews related work. Section 3.2 introduces a rep-

resentation of the pedestrian's history of poses as a random walk on a weighted graph, and Section 3.3 derives a number of entropy and relative entropy metrics as indicators of uncertainty. Finally, Section 3.4 shows experimental results and Section 3.5 discusses the main contributions of this chapter.

3.1 Introduction

3.1.1 Entropy Definitions

Entropy In the information theory domain, (Shannon) entropy is used to characterize the uncertainty about a source of information [140]. For our purpose, it is the expected value of the information contained in any hexagon's local map.

We define **differential entropy** as the entropy of a continuous random variable, simply shortened by *entropy* and we denote it by $\mathrm{h}(\cdot)$ [34].

We define **entropy rate** as the rate of *growth* of the entropy of a discrete stochastic process over time and we denote it by $\mathrm{H}(\cdot)$.

To avoid confusion with variable names, in this chapter we will use index i to refer to the outgoing hexagon (as opposed to h used in Section 2.5) and index j for incoming hexagon.

Relative Entropy We define the relative entropy (or Kullback-Leibler (KL) divergence), $d_{\mathrm{KL}}(p||q)$ between two probability density functions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ of the same random variable $\boldsymbol{x}$ as a measure of the information loss incurred when $q(\boldsymbol{x})$ is used as an approximation of the true distribution $p(\boldsymbol{x})$ [34]. To describe $p(\boldsymbol{x})$ with the approximating distribution $q(\boldsymbol{x})$ we will need:

$$\mathrm{h}(p(\boldsymbol{x})) + d_{\mathrm{KL}}(p(\boldsymbol{x})||q(\boldsymbol{x})) \text{ bits}\,,$$

instead of $\mathrm{h}(p(\boldsymbol{x}))$ bits. With this definition d_{KL} is always greater than 0, and 0 if and only if $p(\boldsymbol{x}) = q(\boldsymbol{x})$ and it can be used to reflect how well $q(\boldsymbol{x})$ explains $p(\boldsymbol{x})$. Note that this function is asymmetric.

In our analysis of the uncertainty of probabilistic FootSLAM maps based on human odometry we can distinguish between map uncertainty and human motion uncertainty.

3.1.2 Map Uncertainty

In FootSLAM, given that we estimate spatial transitions (the map) conditioned on a sequence of observations (the pedestrian's step measurements), maps are uncertain because of the following reasons:

1. Main corridors and some areas of a building such as conference rooms are visited more often than other rooms; likewise, public buildings such as airports or shopping centers are visited more frequently than residential buildings. Therefore, the number of observations is location-dependent and as a result, areas with a lower number of visits will be more uncertain than areas more frequently visited.

2. Step measurements are collected by means of inaccurate sensors. This can lead to errors when mapping spatial transitions.

3. FootSLAM maps are based on a discrete representation of the space (a grid of hexagons), which leads to a quantization of the step direction probabilities (in this case the pedestrian has 6 angular choices at each location) and which limits the spatial resolution to the size of each hexagon.

To measure the map's uncertainty we can turn to Shannon's information theory. This theory, presented in 1948 [140], has inspired several research articles in cartography for quantifying the information contents and uncertainty of a map. Bjørke [20] describes the problems of applying Shannon's theory to cartography. In his work, he focuses on applying Shannon's information theory to the *syntactic* component of a map and defines the terms *map entities* and *map information source*. He distinguishes between *topological entropy*, *metrical entropy* and *positional entropy*. In addition, he points out that one way to deal with spatial correlation within the map is the computation of the weighted mean of the entropy at different orders of neighborhood.

The idea of computing the entropy of a map has also been exploited in robotics [11]. First, a database of optical flow images for different objects are collected by robots. Then, offline, entropy maps are built from those images. Finally, following a sequential Bayesian approach, these maps are used to guide the motion of the robot along the optimal trajectory that minimizes the uncertainty of recognition.

In the field of robotic SLAM, Blanco et al. [21] address the calculation of the uncertainty of maps built with a RBPF-based approach. The problem is similar to FootSLAM's, but instead of using an angular transition probability map, their map representation is based on an occupancy grid: The space is divided into cells (squares) and the binary probability of occupancy/freeness of that cell is stored. To eliminate the influence of the cells not explored by the robot, Blanco et al. propose to calculate the entropy-based information metric for all cells of a map [21]:

$$I = \sum_{x,y} (\mathrm{H}_{\max} - \mathrm{H}_{x,y}), \tag{3.1}$$

where $H_{x,y}$ represents the entropy of the cell at column x and row y and $H_{\max}$ is the maximum cell entropy. In their article, they also compare the map mean information with the joint entropy of the map and the robot path proposed by [148]. It is shown that when computing the joint entropy, the map entropy dominates and the path entropy is hidden. In [148] the entropy of the individual particle maps is calculated, whereas Blanco et al. propose to use the *expected map* - a weighted average of all particle maps of the

RBPF, what we have called posterior or cumulative map - in order to measure also the consistency of the all particle maps.

In this chapter, and similarly to [20], we will apply Shannon's theory to the syntactic component of FootSLAM probabilistic maps (built on a grid of hexagons) and restrict our problem to calculating the positional entropy. Per definition of the positional entropy, all occurrences of the map entities are seen as unique events and spatial correlations are not considered. As proposed in [21], in our experiments, when performing entropy calculations, we will focus on computing the entropy of *cumulative maps* (Section 2.5.5).

3.1.3 Human Motion Uncertainty

This uncertainty represents the degree of unpredictability of human motion. Naturally, it is very difficult if not impossible to model and characterize this process accurately. For instance, intentions will vary depending on personality, culture, season of the year and past experiences at a specific location.

The predictability of human motion has been studied in [147] and more recently in [144]. Based on the mobility patterns of mobile phone users Song et al. [147] show that human mobility is predictable to a 93% [147]. The authors use a combination of the empirically determined user entropy and Fano's inequality to evaluate this predictability. However, their evaluation comprises datasets with very low spatiotemporal resolution and no topological constraints. Smith et al. [144] show that this upper bound for the predictability of human motion can actually decrease almost exponentially as the spatial resolution increases.

In the derivation of FootSLAM we assumed that the probability of a certain human step is only dependent on the current location, in other words, that the human is memory-less, and makes choices based entirely on what her intentions are at each particular location. In the following section, we describe this simple (first order) location dependent model of human motion as a random walk on a weighted graph.

3.2 Pedestrian's History of Poses as a Random Walk on a Weighted Graph

This section presents a novel interpretation of the sequence of pedestrian poses as a random walk on a weighted graph. This provides new insights on the FootSLAM estimation process and will prove useful to derive metrics regarding the uncertainty of a pedestrian's steps in Section 3.3.

The theory of a random walk on a finite network was introduced by Doyle and Snell in [43] further described by Lovasc [103]. More information can be found in [34].

3.2.1 Markov Chain Interpretation

Consider the stochastic process of being located at one of the N_H hexagons in the grid as a result of the sequence of pedestrian poses $\boldsymbol{P}_{0:k}$. We represent this process as $\boldsymbol{H}_{0:k}$. The set of states corresponds to the pedestrian being located at any of the hexagons of the set $H = \{H_0, H_1, \ldots, H_i, H_j \ldots, H_{N_H-1}\}$. Note that we use k as the time index for the random variable $\boldsymbol{H}$ and i, j as spatial indices of the hexagons.

To compute the joint probability over the sequence of pedestrian poses conditioned on the map we can use as a proxy the joint probability over the hexagons at which the pedestrian was located at each time step (see more interpretation remarks in the next section). Remember that in Section 2.5.4 we assumed that the current pose of the pedestrian only depends on his previous pose (i.e. a Markov process of order one). As a result, the conditional distribution of $\boldsymbol{H}_k$ depends only on the value of the previous observation $\boldsymbol{H}_{k-1} = H_i$. Then, the joint probability distribution over the poses conditioned on the map results in a **chain** of conditional probabilities:

$$p(\boldsymbol{P}_{0:k}|\boldsymbol{M}) \equiv p(\boldsymbol{H}_{0:k}) = p(\boldsymbol{H}_k|\boldsymbol{H}_{k-1}) \cdot p(\boldsymbol{H}_{k-1}|\boldsymbol{H}_{k-2}) \cdots p(\boldsymbol{H}_1|\boldsymbol{H}_0) \cdot p(\boldsymbol{H}_0). \tag{3.2}$$

This is called a Markov chain of order one and we denote it as $\mathcal{P}$.

In the same way the pedestrian can start his walk in any of the hexagons and move from one hexagon to the next, the associated Markov chain can start in any of the states and can move successively from one state to the next (Fig. 3.1). Each one of these moves is called *step* or *transition* [34].

In FootSLAM we learn the probability of moving from hexagon H_i to hexagon H_j by counting the number of times a particle crosses the edges of the hexagons it visits (recall (2.61)). This yields:

$$P(\boldsymbol{H}_k = H_j|\boldsymbol{H}_{k-1} = H_i) = p_{ij} = \frac{C_{ij} + \alpha_{ij}}{C_i^T + \alpha_i^T}. \tag{3.3}$$

Since this probability is independent of k (because we assumed that the underlying map $\boldsymbol{M}$ of step direction probabilities is time invariant and it has been already learnt), the Markov chain is said to be *stationary*: we can shift the sequence of poses (or hexagons) by any amount l, $\boldsymbol{P}_{l:k+l}$, and its joint distribution conditioned on the map $p(\boldsymbol{P}_{l:k+l}|\boldsymbol{M})$ remains the same.

Thus, the chain can be defined by the initial distribution $\vec{x}_0 = p(\boldsymbol{H}_0)$ and the so called *transition matrix* $\vec{\mathrm{P}}$. The transition matrix represents all possible transitions from any hexagon (state) in the hexagonal grid (Markov chain) to any other hexagon (state). The transition matrix for our transition map is therefore a $N_H \times N_H$ matrix:

$$\vec{\mathrm{P}} = \begin{pmatrix} p_{0,0} & \cdots & p_{0,N_H-1} \\ \vdots & \ddots & \vdots \\ p_{N_H-1,0} & \cdots & p_{N_H-1,N_H-1} \end{pmatrix} \tag{3.4}$$

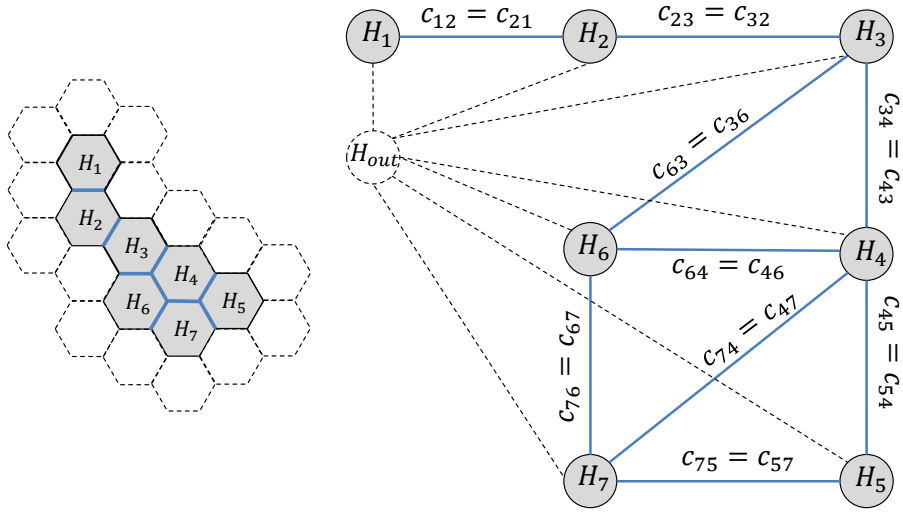


Figure 3.1: Undirected graph (on the right) associated to the FootSLAM transition map on the left. Each visited hexagon (in gray) is related to a state in the graph. The outer hexagons around the visited ones are drawn with a dashed line and represented by the state H_{out}. The edges crossed by the pedestrian are marked in blue and represented in the graph by blue links. The pedestrian can move from one state (hexagon) to another with a probability proportional to the weights c_{ij}, shown on next to each link.

Clearly, each row of the transition matrix will consist of six elements unequal to zero at the connecting hexagons – and zeros for the remaining elements. Note that we assume that with every step the pedestrian takes, he is located at a different hexagon, i.e. $p_{ii} = 0$. Furthermore, any Markov transition matrix has to fulfill the condition

$$\sum_j p_{ij} = 1, \quad \forall\, j = 0, \ldots, N_H - 1\,.$$

Given the transition matrix $\vec{\mathrm{P}}$ and the state distribution at time $\vec{x}_{k-1} = p(\boldsymbol{H}_{k-1})$, computing state probability distribution at time k is as simple as:

$$\vec{x}_k = \vec{x}_{k-1}\vec{\mathrm{P}}. \tag{3.5}$$

After n steps, the entries p_{ij}^n of $\vec{\mathrm{P}}^n$ (the transition matrix raised to the n-th power) represent the probability that the pedestrian, initially located at H_i will move to H_j in n steps.

Markov Chain Properties

FootSLAM maps consist of a set of visited hexagons with their corresponding hexagon transition counters (recall (2.63)). However, the pedestrian can leave the mapped area

with a positive probability (thanks to the prior counts represented by α terms in (2.61)), making our transition map virtually infinite. Nevertheless, we can assume that the transition map lies on a globe (Fig. 3.2) so that the associated Markov chain is finite. Proceeding like this, the following properties hold:

- The grid of hexagons has no borders but yet it is **finite**, i.e. the Markov chain is finite. Furthermore, we can study the subset of states of the Markov chain corresponding to the hexagons that the pedestrian visited (in dark gray in Figs. 3.1 and 3.2).
- The Markov chain represented by the grid of hexagons is **irreducible** (also called ergodic): it is possible to reach any hexagon from any other hexagon in a finite number of steps.
- The Markov chain is **aperiodic**, i.e. the states do not return to the starting position with a certain period.
- The Markov chain is **recurrent**, i.e. with a probability of one it is possible to return to a hexagon within finite time.

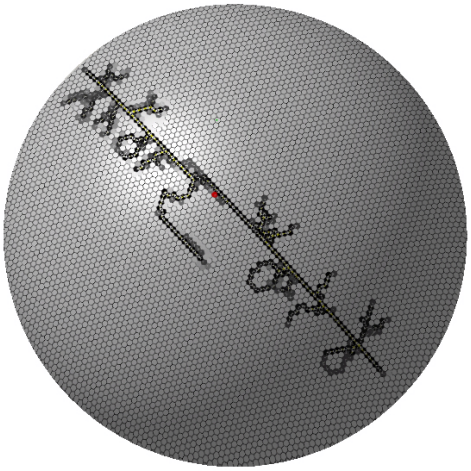


Figure 3.2: Globe representation of a FootSLAM map obtained in an office environment. The hexagons visited by the pedestrian are marked in darker colors.

For any irreducible and aperiodic Markov chain with transition matrix $\vec{\mathrm{P}}$, there exists a unique stationary distribution $\vec{\pi} = \{\pi_i\}\ \forall\, i$ s.t. $H_i \in H$ that is independent from the initial distribution $\vec{x}_0$ of the chain and that satisfies [37][34]:

$$\vec{\pi}\,\vec{\mathrm{P}} = \vec{\pi}\,. \tag{3.6}$$

3.2.2 Random Walk on a Weighted Graph

A network is defined as an undirected graph G with a set of vertices (also called nodes) corresponding to the states and a set of links (also called edges or branches) that connect

the nodes. Each link is associated to a non-negative weight $c_{ij} = c_{ji}$, usually called *conductance* in electric networks. A random walk on G is a Markov chain with the elements of the transition matrix defined as [34][121]:

$$p_{ij} = \frac{c_{ij}}{c_i^T}, \tag{3.7}$$

where $c_i^T = \sum_j c_{ij}$.

Comparing (3.3) with (3.7) it is easy to verify that the random walk $\boldsymbol{P}_{0:k-1} = \mathrm{P}_{0:k-1}$ conditioned on the map $\boldsymbol{M}$ corresponds to a sequence of nodes in a graph, i.e. hexagons in the grid, and that the links between the hexagons have a weight defined as $c_{ij} = C_{ij}+\alpha_{ij}$.

Note that $c_{ji} = c_{ji}$ for any pair of neighboring hexagons, since $C_{ij} = C_{ji}$ (remember we increment both C_{ij} and C_{ji} when crossing edge e_{ij}, see Fig. 2.10(c)) and $\alpha_{ij} = 0.8\ \forall\, i, j$.

If we define:

$$c_G = \sum_{i=0}^{N_{H-1}} c_i^T = \sum_{i=0}^{N_{H-1}} (C_i^T + \alpha_i^T) = C_M + \alpha_M,$$

with $C_M = \sum_{i=0}^{N_{H-1}} C_i^T$ and $\alpha_M = \sum_{i=0}^{N_{H-1}} \alpha_i^T$, and:

$$\pi_i = \frac{c_i^T}{c_G} = \frac{C_i^T + \alpha_i^T}{C_M + \alpha_M}, \tag{3.8}$$

such a Markov chain has the additional property that it is reversible with respect to the probability $\vec{\pi}$, i.e. :

$$\pi_i\ p_{ij} = \pi_j\ p_{ji}, \quad \forall\, i, j\,. \tag{3.9}$$

As a result, $\vec{\pi}$ represents the **unique stationary distribution** for the reversible Markov chain [37].

Our goal in this chapter is to derive a metric to measure the uncertainty of a pedestrian's steps and the uncertainty of a FootSLAM map. For this purpose, we now introduce the concepts of map entropy, step entropy rate, relative map entropy and relative step entropy rate within the context of FootSLAM.

3.3 FootSLAM Entropy Metrics

In this section we are interested in deriving metrics to measure the following uncertainties:

1. The uncertainty of the map (Fig. 3.3(a)). Since a hexagon's local map follows a (continuous) Dirichlet distribution, we propose using the differential entropy to measure its uncertainty. This metric reflects how certain the learned transition probabilities at each hexagon are.

2. The uncertainty of the pedestrian's steps (Fig. 3.3(b)), which is influenced by the interplay between the constraints of the environment and the pedestrian's choice of movement therein. As shown in Section 3.2, the sequence of pedestrian's steps can be interpreted as a discrete stochastic process, with each step placing the pedestrian at a different hexagon. As such, we shall measure its uncertainty based on its entropy rate.

3.3.1 Proposed Entropy Metrics and Interpretation

The map uncertainty and step direction uncertainty can be computed based on:

i) The history of noisy measurements $\boldsymbol{Z}^U_{1:k} = Z^U_{1:k}$, where $Z^U_{1:k}$ is a given measurement sequence.

ii) The history of poses $\boldsymbol{P}_{0:k} = P_{0:k}$, where $P_{0:k}$ is true pose sequence. This is unknown to us but we can suppose that we have a genie who can tell us about the true pose.

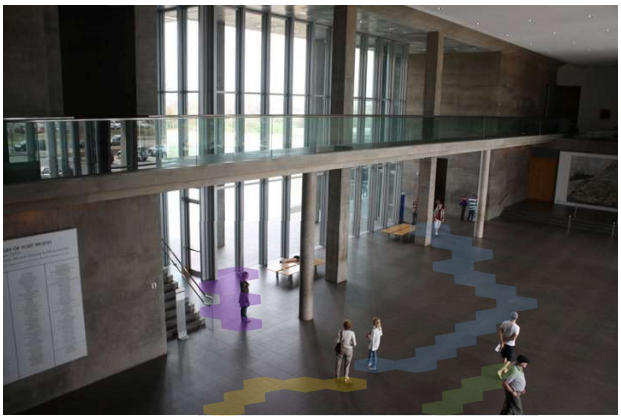
iii) The true map $\boldsymbol{M} = M$, also given by a genie.

For ease of notation in the following derivations we will omit the step vector $\boldsymbol{U}_k$, since knowledge of $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k-1}$ gives us knowledge of the step vector $\boldsymbol{U}_k$. Also note that we will omit the particle index m or the superindex cum when calculating entropy metrics since all the equations apply for both a particle's transition map and the cumulative map.

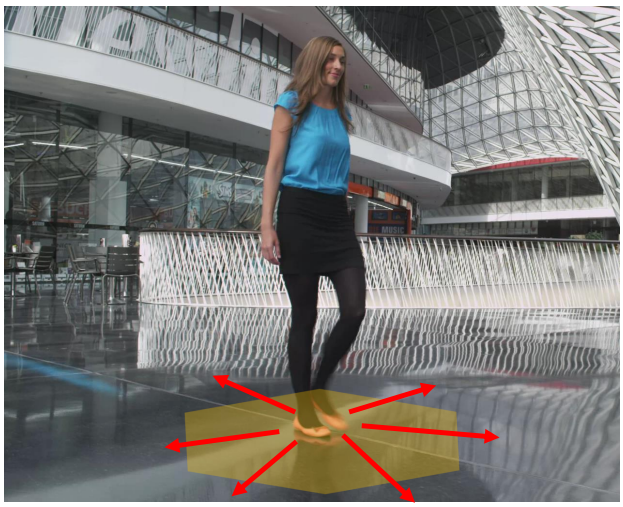
An overview of the different kinds of entropy metrics is provided in Table 3.1. Similar reasoning can be applied to the relative entropy between two map instances inferred from two pose sequences in the same environment and to the relative entropy rate of the pedestrian's steps between two pose sequences in the same environment.

	map entropy	step entropy rate	source
convergence $\Downarrow$	$\mathrm{h}(p(\boldsymbol{M}\|\boldsymbol{Z}^U_{1:k} = Z^U_{1:k}))$	$\lim_{k\to\infty} \mathrm{H}(\boldsymbol{Z}^U_k\|\boldsymbol{Z}^U_{1:k-1} = Z^U_{1:k-1})$	FootSLAM
infinite observations $\Downarrow$	$\mathrm{h}(p(\boldsymbol{M}\|\boldsymbol{P}_{0:k} = P_{0:k}))$	$\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k\|\boldsymbol{P}_{0:k-1} = P_{0:k-1})$	genie pose
	$\mathrm{h}(p(\boldsymbol{M}\|\boldsymbol{M} = M)) \to$ no uncertainty	$\mathrm{H}(\boldsymbol{P}_k\|\boldsymbol{P}_{k-1} = P_{k-1}, \boldsymbol{M} = M)$	genie map

Table 3.1: Overview of entropy metrics. We distinguish between map entropy and step entropy rate based on the sequence of measurements $Z^U_{1:k}$, on the sequence of true poses $P_{0:k}$, or the true map M. On the left side, $\Downarrow$ indicates the following: if FootSLAM converges, the metrics based on the sequence of step measurements become equivalent to the metrics based on the true pose; if in addition we have infinite observations, this leads to a map with no uncertainty and the step entropy rate would encompass the remaining uncertainty of the next step of the pedestrian.



(a) People walking at the entrance of a museum (Tadao Ando Museum, Fort Worth, TX). Each color represents the hexagons visited by each pedestrian corresponding to their history of poses. The map entropy represents the average uncertainty of the learned transition probabilities at each hexagon.



(b) A pedestrian chooses her next step based on her current location, her visual cues, her intentions and the constraints of the environment. The arrows indicate the possible directions she can take to exit the hexagon where she is currently located. The step entropy rate measures the average uncertainty regarding her next step.

Figure 3.3: Illustration of map entropy and step entropy rate.

FootSLAM Convergence

In our subsequent numerical evaluations we cannot resort to a genie, hence we shall assume that FootSLAM achieves good convergence and we can therefore approximate the output of the genie using the *estimated* history of poses by FootSLAM.

To be specific, we define perfect convergence of FootSLAM if the estimated pose sequence is equal to the (true) genie-sequence in the sense that the genie-sequence and the FootSLAM sequence lead to identical estimates of the map. This is the same as saying that the estimated sequence might still be perfect in this sense, despite small deviations from the genie-sequence, if all map relevant information is identical, e.g. leads to the same discrete transitions in the map.

If perfect convergence is achieved then we are able to determine the map entropy $\mathrm{h}(p(\boldsymbol{M}|\boldsymbol{P}_{0:k} = P_{0:k}))$ and the step entropy rate $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|\boldsymbol{P}_{0:k-1} = P_{0:k-1})$. In the following sections we refer to the entropy metrics conditioned on the estimated history of poses, assuming sufficiently good convergence.

Infinite Observations

The entropy metrics conditioned on the *true map* (last row of Table 3.1) are metrics that we are not able to calculate since the map itself is being estimated from finite and not necessarily representative walks.

If we had the true map, the map entropy given the true map would obviously result in no remaining uncertainty. On the other hand, the step entropy rate given the true map $\mathrm{H}(\boldsymbol{P}_k|\boldsymbol{P}_{k-1} = P_{k-1}, \boldsymbol{M} = M)$ would encompass the remaining uncertainty of the next step of the pedestrian that is only influenced by the pedestrian's free choice within the perfectly known physical constraints and other physical features of the environment. Conceptually one can imagine numerous ways to quantify the statistical influence of the environment on human motion. Obstacles and constraints obviously physically enforce limitations on motion, but features such as signs, desks, items on display in a museum, semantically relevant features such as a podium in a conference room, all couple with the human cognitive system and influence motion. In addition, it is to be expected that such influencing factors are dependent on the individual, and the state of the individual and her context (such as travel plans, etc). Since we have no observation of the physical environment, we shall rely on the human pose sequence as the only state that we can use to encode the environment's influence on motion. This is equivalent to saying that *an infinitely long sequence of representative poses $P_{0:\infty}$ is as informative about the person's next step as the perfect map.*

As a result, the step entropy rate will be assumed to be constant over time and the step entropy rate given the genie history of poses $\boldsymbol{P}_{0:k-1} = P_{0:k-1}$ approaches asymptotically this constant for $k \to \infty$:

$$\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|\boldsymbol{P}_{0:k-1} = P_{0:k-1}) = \mathrm{H}(\boldsymbol{P}_k|\boldsymbol{P}_{k-1} = P_{k-1}, \boldsymbol{M} = M)\,.$$

The next subsections show the mathematical derivation of these metrics.

3.3.2 Map Entropy Conditioned on the History of Poses

In this section we address the computation of the uncertainty of the map $\boldsymbol{M}$ conditioned on the history of poses $\boldsymbol{P}_{0:k} = \mathrm{P}_{0:k}$, denoted $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}_{0:k}))$. Note that in the following for convenience we abbreviate the condition $\boldsymbol{P}_{0:k} = \mathrm{P}_{0:k}$ with $\mathrm{P}_{0:k}$. Map entropy $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}_{0:k}))$ takes into account how many times a transition of the map is measured and how certain a learned transition probability is. The map entropy can be calculated as the average of the hexagon entropies [34]:

$$\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}_{0:k})) = \frac{1}{N_H} \sum_{i=0}^{N_H-1} \mathrm{h}(p(\boldsymbol{M}_i|\mathrm{P}_{0:k}))\,, \tag{3.10}$$

given that the discretization into a grid of independent hexagons renders the local maps $\boldsymbol{M}_i$ independent of one another (2.50).

To compute the entropy of a hexagon's local map $p(\boldsymbol{M}_i|\mathrm{P}_{0:k})$ we rely on the assumption that $p(\boldsymbol{M}_i|\mathrm{P}_{0:k})$ follows a Dirichlet distribution, which is a continuous distribution (see Section 2.5.4). The formula for the entropy of a continuous random variable $\boldsymbol{x}$, usually called *differential* entropy is [34]:

$$\mathrm{h}(p(\boldsymbol{x})) = -\int_{\boldsymbol{x}=-\infty}^{\boldsymbol{x}=\infty} p(\boldsymbol{x}) \log p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{3.11}$$

If the logarithm has 2 as base, then the entropy is expressed in bits. In this chapter, we will use the unit bits. Note that the range of the entropy defined in this manner is $(-\infty, 0]$ bits. The lower the entropy value, the lower the uncertainty about the random variable.

In our case $p(\boldsymbol{x}) = p(\boldsymbol{M}_i|\mathrm{P}_{0:k})$ follows a Dirichlet distribution (2.56) with parameters $u_i = \{u_{ij}\} = \{C_{ij} + \alpha_{ij}\}\ \forall\, j \mapsto i$, where the symbol $\mapsto$ indicates that hexagon H_i and hexagon H_j share a connecting edge. As a result, the differential entropy has the following closed-form [9]:

$$\begin{aligned} \mathrm{h}(p(\boldsymbol{M}_i|\mathrm{P}_{0:k})) &= \log \mathrm{B}(C_i^T + \alpha_i^T) + (C_i^T + \alpha_i^T - 6)\Psi(C_i^T + \alpha_i^T) \\ &- \sum_{j \mapsto i} (C_{ij} + \alpha_{ij} - 1)\Psi(C_{ij} + \alpha_{ij})), \end{aligned} \tag{3.12}$$

with the same Beta function defined in (2.57) and Gamma function of (2.58) and Digamma function defined as:

$$\Psi(x) = \frac{\mathrm{d}}{\mathrm{d}x} \ln \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}\,. \tag{3.13}$$

The maximum entropy of the map (obtained by inserting (3.12) in (3.10)) occurs for $C_{ij} + \alpha_{ij} = 1\ \forall\, j \mapsto i$, and has a value of $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}_{0:k})) = -4.79$ bits.

3.3.3 Entropy Rate of the Pedestrian's Steps

Consider now the Markov chain $\mathcal{P}$ of the sequence of pedestrian poses $\boldsymbol{P}_{0:k} = \mathrm{P}_{0:k}$. The appropriate question to address is the *rate* of growth of its entropy with k. The entropy rate of a stochastic process $\mathcal{X}$ is defined as [34]:

$$\mathrm{H}(\mathcal{X}) = \lim_{k\to\infty} \frac{1}{k}\mathrm{H}(\boldsymbol{X}_{0:k}). \tag{3.14}$$

The conditional entropy H'($\mathcal{X}$) of a stochastic process $\boldsymbol{X}_{0:k}$ is defined as [34]:

$$\mathrm{H'}(\mathcal{X}) = \lim_{k\to\infty} \mathrm{H}(\boldsymbol{X}_k|\boldsymbol{X}_{0:k-1}). \tag{3.15}$$

For a **stationary** stochastic process, both limits exist and they are equal to: $\mathrm{H'}(\mathcal{X}) = \mathrm{H}(\mathcal{X}) = \mathrm{H}(\boldsymbol{X}_1|\boldsymbol{X}_0)$, which for a Markov chain $\mathcal{P}$ with stationary distribution $\vec{\pi}$ and transition matrix $\vec{\mathrm{P}}$ yields [34]:

$$\mathrm{H}(\mathcal{P}) = \lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|\boldsymbol{P}_{0:k-1}) = \sum_{i=0}^{N_H-1} \pi_i \sum_{i\mapsto j} \mathrm{H}(p_{ij}), \tag{3.16}$$

with:

$$\mathrm{H}(p_{ij}) = -p_{ij}\log p_{ij}\,. \tag{3.17}$$

By inserting (3.7) and (3.8) in (3.16) we obtain:

$$\mathrm{H}(\mathcal{P}) = -\sum_{i=0}^{N_H-1} \frac{C_i^T + \alpha_i^T}{C_M + \alpha_M} \sum_{j\mapsto i} \frac{C_{ij}+\alpha_{ij}}{C_i^T+\alpha_i^T} \log \frac{C_{ij}+\alpha_{ij}}{C_i^T+\alpha_i^T}\,. \tag{3.18}$$

The entropy rate of the sequence of pedestrian poses provides us with the stationary growing rate of the step entropy, i.e., the uncertainty regarding each next step. In other words, it represents the average entropy of the next move from H_i to H_j out of each hexagon $H_i \in H$, with the average weighted by the stationary probability of being in state H_i.

The maximum entropy rate occurs with an equally probable Markov chain ($\pi_i = 1/N_H \;\forall\, i$ s.t. $H_i \in H$) with uniform transition probabilities ($p_{ij} = 1/6 \;\forall\, j \mapsto i$) and has a value of $\mathrm{H}(\mathcal{P}) = 2.584$ bits.

3.3.4 Relative Map Entropy Conditioned on the History of Poses

The relative entropy between two distributions p and q of a continuous random variable $\boldsymbol{x}$ is defined as [34]:

$$d_{\mathrm{KL}}(p||q) = \int_{-\infty}^{\infty} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\,\mathrm{d}\boldsymbol{x}\,. \tag{3.19}$$

Suppose we have two sequences of pedestrian poses $\mathrm{P}^P_{0:k}$ and $\mathrm{P}^Q_{0:k}$, each yielding an estimate of the same map, M^P and M^Q respectively. We define $p = p(\boldsymbol{M}|\mathrm{P}^P_{0:k})$ and $q = p(\boldsymbol{M}|\mathrm{P}^Q_{0:k})$ and our goal is to compute the relative entropy between them.

Similarly to the derivation of the map entropy conditioned on the history of poses in Section 3.3.2, to compute the map relative entropy we rely on the assumption that the local maps at each hexagon are independent of each other, thus yielding:

$$d_{\mathrm{KL}}(p||q) = \sum_{i \in H_{M^P}} d_{\mathrm{KL}}(p_i||q_i)\,, \tag{3.20}$$

where $p_i = p(\boldsymbol{M}_i|\mathrm{P}^P_{0:k})$, $q_i = p(\boldsymbol{M}_i|\mathrm{P}^Q_{0:k})$ and H_{M^P} is the set of hexagons in M^P.

Furthermore, we assume that p_i and q_i follow each a Dirichlet distribution with parameters $\{u_{ij}\} = \{C^P_{ij} + \alpha^P_{ij}\}$ and $\{v_{ij}\} = \{C^Q_{ij} + \alpha^Q_{ij}\}$, respectively. Then resolving (3.20) for a Dirichlet distribution yields [117]:

$$\begin{aligned} d_{\mathrm{KL}}(p||q) = &\sum_{i \in H_{M^P}} \log \frac{\Gamma(C^{TP}_i + \alpha^{TP}_i)}{\Gamma(C^{TQ}_i + \alpha^{TQ}_i)} + \sum_{j \mapsto i} \log \frac{\Gamma(C^P_{ij} + \alpha^P_{ij})}{\Gamma(C^Q_{ij} + \alpha^Q_{ij})} \\ &+ \sum_{j \mapsto i} \left(C^Q_{ij} + \alpha^Q_{ij} - C^P_{ij} - \alpha^P_{ij}\right) \cdot \left(\Psi(C^Q_{ij} + \alpha^Q_{ij}) - \Psi(C^{TQ}_i + \alpha^{TQ}_i)\right)\,, \end{aligned} \tag{3.21}$$

with the Gamma and Digamma function definitions in (2.58) and (3.13), respectively.

This equation of the relative map entropy measures the difference between two distributions of the same map and therefore can be used on a hexagon basis to detect similarities between two maps.

3.3.5 Relative Entropy Rate of the Pedestrian's Steps

Suppose that $\mathcal{P}$ and $\mathcal{Q}$ are two Markov chains representing two different pose sequences $\mathrm{P}^P_{0:k}$ and $\mathrm{P}^Q_{0:k}$, respectively. Each one of those pose sequences results in a FootSLAM map, namely, M^P and M^Q. If their transition matrices are $\vec{\mathrm{P}}$ and $\vec{\mathrm{Q}}$, respectively, we can compute the relative step entropy rate (or divergence rate, dr) as follows [121]:

$$\mathrm{dr}_{\mathrm{KL}}(\mathcal{P}||\mathcal{Q}) = \lim_{k\to\infty} \frac{1}{k} d_{\mathrm{KL}}(p(\boldsymbol{P}_k|\mathrm{P}^P_{0:k-1})||p(\boldsymbol{P}_k|\mathrm{P}^Q_{0:k-1})) = \sum_{i \in H_{M^P}} \pi^P_i \sum_{j \mapsto i} p_{ij} \log \frac{p_{ij}}{q_{ij}}\,, \tag{3.22}$$

where $\vec{\pi}^P = \{\pi^P_i\}$ is the unique stationary distribution of $\vec{\mathrm{P}}$.

Inserting (3.7) and (3.8) in (3.22) we obtain:

$$\mathrm{dr}_{\mathrm{KL}}(\mathcal{P}||\mathcal{Q}) = \sum_{i \in H_{M^P}} \frac{C^{TP}_i + \alpha^T_i}{C^P_M + \alpha_M} \sum_{j \mapsto i} \frac{C^P_{ij} + \alpha_{ij}}{C^{TP}_i + \alpha^T_i} \log \frac{\dfrac{C^P_{ij} + \alpha_{ij}}{C^{TP}_i + \alpha^T_i}}{\dfrac{C^Q_{ij} + \alpha_{ij}}{C^{TQ}_i + \alpha^T_i}}\,, \tag{3.23}$$

where the superindices P and Q in the counters indicate that they belong to map M^P and map M^Q, respectively, and H_{M^P} represents the set of hexagons visited by $\mathrm{P}^P_{0:k}$.

The relative step entropy rate between two pose sequences describes how well the map constructed as a result of $P^P_{0:k}$, i.e. M^P, explains the uncertainty regarding the steps of the pose sequence $P^Q_{0:k}$.

In the next section we apply these entropy metrics to FootSLAM maps obtained by procesing real odometry data collected in an indoor environment.

3.4 Experimental Results

3.4.1 Experiment Settings

Two pedestrians collected a total of 6 datasets $D_1, \ldots, D_6$ within the same office environment (depicted in Fig. 3.4). The pedestrians walked between 5 and 25 minutes wearing a foot mounted IMU (MTx - Xsens®). The data were preprocessed by a step estimator based on ZUPTs and an UKF [166] and then used as input for the FootSLAM algorithm, which was executed offline. In the estimation process, the radius of the hexagons was set to $r = 0.5\,\mathrm{m}$ and the number of particles used in the RBPF was $N_p = 10\,000$. The resulting posterior maps are denoted as $M_1, \ldots, M_6$.

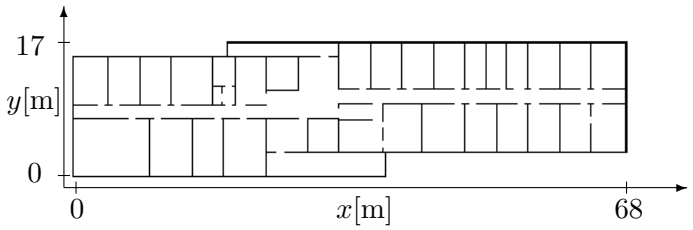
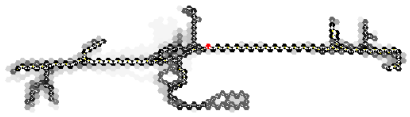


Figure 3.4: Floor plan of the second floor of an office building.

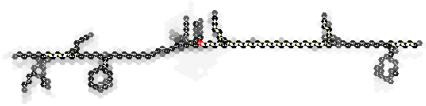
Dependency of Entropy Metrics with the Size of the Map

The map entropy and step entropy rate equations ((3.10) and (3.16), respectively) depend on the number of hexagons N_H in the grid. However, adding the entropy of the hexagons that have not been visited by a walk (each with maximum entropy corresponding to equiprobable crossings of their edges) hides the entropy values of the hexagons that were visited during the walk.

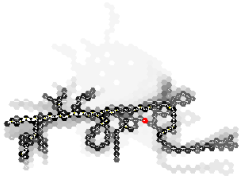
To cope with this problem we calculate the map entropy and step entropy rate by summing over all the hexagons contained in any of the maps we wish to compare, but excluding all the hexagons that have not been visited (the rest of the world).



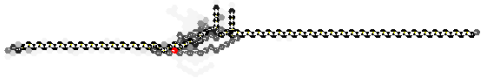
(a) Map M_1: $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^1_{0:k})) = -5.11$ bits, $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^1_{0:k-1}) = 2.51$ bits



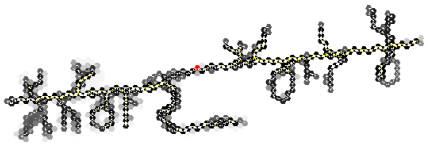
(b) Map M_2: $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^2_{0:k})) = -5.08$ bits, $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^2_{0:k-1}) = 2.52$ bits



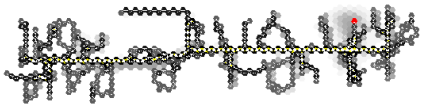
(c) Map M_3: $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^3_{0:k})) = -5.02$ bits, $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^3_{0:k-1}) = 2.54$ bits



(d) Map M_4: $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^4_{0:k})) = -5.10$ bits, $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^4_{0:k-1}) = 2.50$ bits



(e) Map M_5: $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^5_{0:k})) = -5.32$ bits, $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^5_{0:k-1}) = 2.45$ bits



(f) Map M_6: $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^6_{0:k})) = -5.33$ bits, $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^6_{0:k-1}) = 2.43$ bits

Figure 3.5: Posterior maps corresponding to the six collected datasets D_1-D_6. The starting position of the pedestrian is marked in red. The corresponding map entropy $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^d_{0:k}))$ and the step entropy rate values $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^d_{0:k-1})$ of the maps M_1 to M_6 are shown in each caption.

3.4.2 Map Entropy and Step Entropy Rate for Map Quality Assessment

The map entropy $\mathrm{h}(p(\boldsymbol{M}|\mathrm{P}^d_{0:k}))$ values and step entropy rate values $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|\boldsymbol{P}_{0:k-1} = P^d_{0:k-1})$ values of the maps M_1 to M_6 are shown in the captions of Fig. 3.5. The superindex d indicates the dataset number.

Map M_6 has the lowest map entropy and step entropy rate. The entropy values of M_6 are closely followed by M_5, and M_3 presents the highest entropy, reflecting the uncertainty over the large unvisited areas. From visual inspection of Fig. 3.5 one expected such results: map M_6 is the most extensive and accurate and M_3 the least.

Note that a higher map entropy value is also reflected in a higher step entropy rate value. However, one advantage of using the map entropy over the step entropy to asses the goodness of a map is that the map entropy reflects the number of visits per hexagon. Maps corresponding to walks that revisit areas more often results in lower entropy.

To illustrate that, we show the results for the differential entropy of the map M_6 for different *map strengthening factors* in Fig. 3.6: The transition counters of the map are multiplied with the map strengthening factor, simulating a higher number of visits. For strengthening factors greater than 100, the differential entropy decreases linearly with the logarithm of the strengthening factor despite the fact that we only increased the total counters of each hexagon and did not change relationship of the edge transition probabilities.

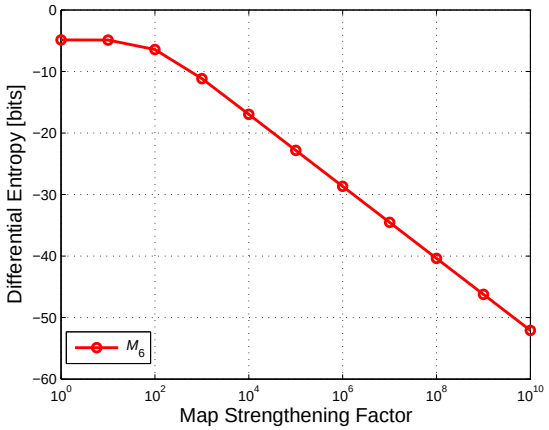


Figure 3.6: Differential map entropy of the map M_6 (Fig. 3.5(f)) conditioned on the history of poses (3.12) for different map strengthening factors. For a map strengthening factor greater than 100, the differential entropy decreases linearly with the logarithm of the strengthening factor.

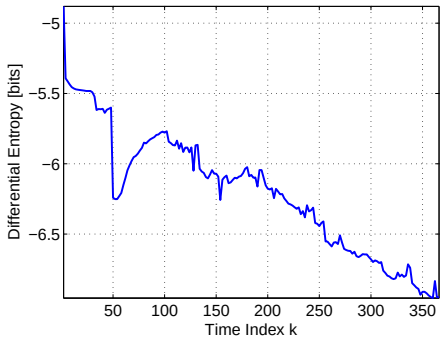
We propose to use the map entropy as a metric to choose the order in which N_d maps are combined in the iterative "Turbo" FeetSLAM algorithm (see Section 2.6.2): we shall order the maps by increasing entropy and start combining the maps with the lowest entropy until we have one single combined map. This effectively reduces the complexity of the combination step from $\mathcal{O}(N_d^2)$ to $\mathcal{O}(N_d)$.

3.4.3 Map Entropy and Step Entropy Rate over Time

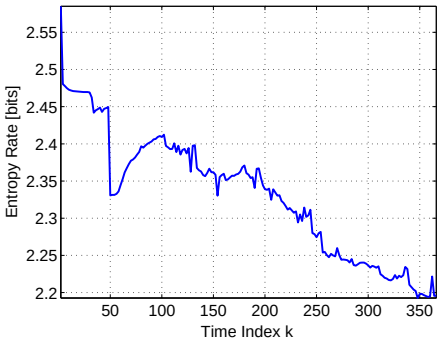
We have illustrated the behavior of the map differential entropy (Fig. 3.7(a)) and the step entropy rate (Fig. 3.7(b)) over time for dataset D_4. For visual comparison, Figure 3.8 shows the estimated posterior map (in gray) and that of the particle with the highest likelihood (in blue) during the FootSLAM estimation process. We show snapshots for time indices k equal to 48, 50, 88, 128, and 154. The entropy calculations only take into account the visited hexagons at each time index.

Before time index $k = 50$ (Fig. 3.8(a)) the pedestrian visits half of the corridor and walks back to his starting position. At time index $k = 50$ the estimations converge (Fig. 3.8(b)) which is reflected in the high decrease of entropy. After this, new areas being explored

results in an increase of the entropy. This is shown in Fig. 3.8(c), which shows a high diversity of hypothesis (they light gray fan-like shape). When revisiting again parts of the corridor, some particle map hypotheses are rewarded and others are killed due to resampling (e.g. time index $k = 128$ and $k = 154$, Fig. 3.8(d) and 3.8(e), respectively), with a sudden decrease in entropy. One can see from the figures that the small jumps in the curves reflect very well the exploring and convergence events.



(a) Differential map entropy for dataset D_4 over time $\mathrm{h}(p(\boldsymbol{M}|P^4_{0:k}))$. At the beginning the entropy is maximum: -4.79 bits.



(b) Step entropy rate $\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k|P^4_{0:k-1})$ for dataset D_4 over time. At the beginning the entropy is maximum: 2.58 bits.

Figure 3.7: Differential entropy and entropy rate for dataset D_4 over time. The visual representations of the posterior map at different time indices are shown in Fig. 3.8. At time index $k = 50$ the estimations converge (Fig. 3.8(b)) which is reflected in the high decrease of entropy at this time index. After that, new area is explored resulting in an increase before it decreases again. The jumps reflect very well the exploring and convergence situations.

3.4.4 KL Divergence between Two Maps

In this section we show the KL divergence between two maps of the same environment: M_6 (Fig. 3.5(f)) and M_5 (Fig. 3.5(e)) resulting from two pose sequences, $\mathrm{P}^6_{0:k}$ and $\mathrm{P}^5_{0:k}$, respectively, $d_{\mathrm{KL}}(p(\boldsymbol{M}|\mathrm{P}^6_{0:k})||p(\boldsymbol{M}|\mathrm{P}^5_{0:k}))$. To illustrate the behavior of the KL distance, we will apply a series of geometric transformations to M_5:

- Spatial translations along the x axis,
- Spatial translations along the y axis,
- Rotations about the starting pose,

followed by a projection onto the same coordinate system [136] and compute the KL divergence between M_6 and M_5 transformed (Fig. 3.9).

The minimum relative entropy is reached for no translation or rotation (as expected, because the maps were initially aligned), indicating that the steps in $P^5_{0:k}$ are explained

well by the transition probabilities of M_6. Whereas the KL divergence grows slowly with the shifts along the x-axis (Fig. 3.9(a)) – given that the maps still share a portion of the corridor (with similar spatial transitions) —, the curve is more steep for translations along the y-axis (Fig. 3.9(b)), which quickly make the two maps not share the main horizontal corridor.

Finally, Fig. 3.9(c) shows the KL divergence between M_5 rotated about the starting hexagon (in red) and M_6 unchanged. The minimum is reached when the two maps are aligned (M_5 unchanged), but we can see that towards the rotation value of π radians there is a local minimum due to the symmetries in the building, which make the left side of the corridor fall onto the right side of the corridor and vice-versa.

Similar results were obtained for the relative entropy rate of the pedestrian's step (3.23).

These results show how the relative map entropy and relative step entropy rate are valid metrics to express how well the transition probabilities of one map can explain those of another map. These results suggest that we can use the relative map entropy between two maps and relative step entropy rate to find the best geometric transformation that places the two maps within the same coordinate system as an alternative for the log-likelihood-based correlation function (2.71), initially proposed in [136].

3.5 Discussion

FootSLAM maps are a probabilistic representation of human motion and, as such, they are uncertain. In this chapter we have differentiated between the uncertainty of the map itself and the uncertainty regarding human motion. To better grasp human motion uncertainty we have introduced a new interpretation of a pedestrian's sequence of steps conditioned on a map as a random walk on a weighted graph, expanding our understanding regarding the inner workings of the FootSLAM estimation process.

We have introduced four entropy-based metrics to measure the uncertainty of the map and the uncertainty of the pedestrian's steps. Table 3.2 summarizes the main insights derived from the experiments. The most impactful result is that we can use the map entropy to reduce the complexity of the "Turbo" FeetSLAM algorithm from quadratic to linear in the number of datasets if we sort the maps by increasing uncertainty.

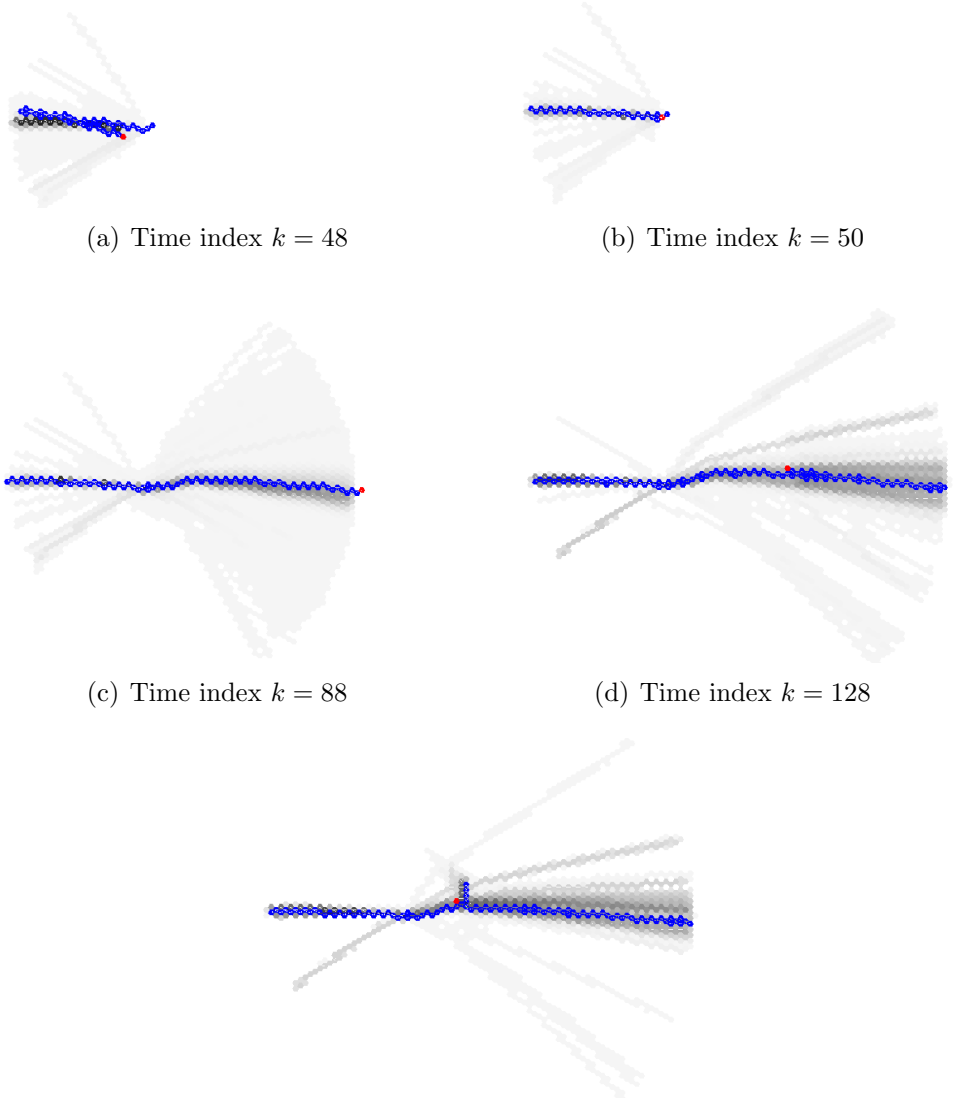
Future work should address the following topics:

- Analyze the correlation between the accuracy of map-aided navigation based on a FootSLAM map and the measured uncertainty of the map: Do maps with lower uncertainty provide better localization accuracy?
- Measure and analyze the uncertainty of a single particle's map or the uncertainty of the pose estimates of the particle cloud over time.
- Extend the presented entropy metrics to multistory environments.

Name	**Formula**	**Remarks**
Map Entropy	$\mathrm{h}(p(\boldsymbol{M}\|P_{0:k}))$	Reflects the inherent uncertainty of the map. It reflects the number of visits of the hexagons. It can be used to select the order in which maps are combined in FeetSLAM, reducing its complexity from $\mathcal{O}(N_d^2)$ to $\mathcal{O}(N_d)$.
Step Entropy Rate	$\lim_{k\to\infty} \mathrm{H}(\boldsymbol{P}_k\|P_{0:k-1})$	Reflects the uncertainty regarding the pedestrian's next step. It does not reflect the number of visits of the hexagons.
Relative Map Entropy	$d_{\mathrm{KL}}(p(\boldsymbol{M}\|\mathrm{P}^P_{0:k})\|\|$ $p(\boldsymbol{M}\|\mathrm{P}^Q_{0:k}))$	Describes how well each hexagon's local map distribution explains that of another map. It can be used to find the geometric transformation that places two maps within the same coordinate system.
Relative Step Entropy Rate	$\lim_{k\to\infty} 1/k\ d_{\mathrm{KL}}(p(\boldsymbol{P}_k\|\mathrm{P}^P_{0:k-1})\|\|$ $p(\boldsymbol{P}_k\|\mathrm{P}^Q_{0:k-1}))$	Describes how well the map learnt as a result of $P^P_{0:k}$, M^P, explains the steps of the pose sequence $P^Q_{0:k}$. It can also be used to find the geometric transformation that places two maps within the same coordinate system.

Table 3.2: FootSLAM entropy metrics.

- Investigate if these entropy metrics can be used to measure the convergence rate of FootSLAM.



(a) Time index $k = 48$

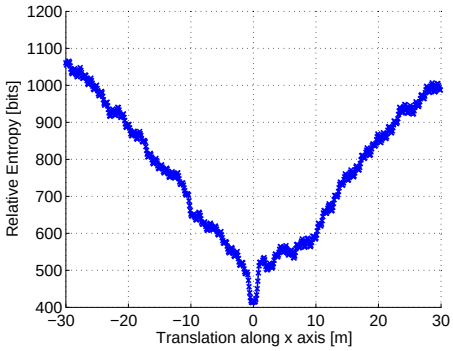
(b) Time index $k = 50$

(c) Time index $k = 88$

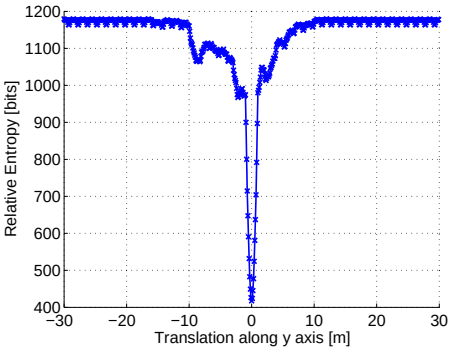
(d) Time index $k = 128$

(e) Time index $k = 154$

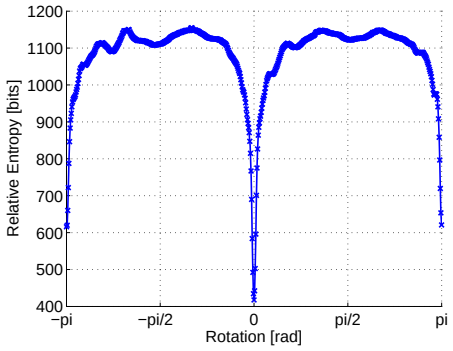
Figure 3.8: Cumulative posterior map obtained by FootSLAM estimation at time indices k equal to 48, 50, 88, 128, and 154. Different shades of gray encode the weight of the corresponding particle (darker means higher weight). The current position is marked in red. The posterior map of the most likely particle is shown in blue.



(a) Relative entropy between two maps M_6 (unchanged) and M_5 translated along the x-axis.



(b) Relative entropy between two maps M_6 (unchanged) and M_5 translated along the y-axis.



(c) Relative entropy between two maps M_6 (unchanged) and M_5, where M_5 is rotated about its starting position.

Figure 3.9: Relative entropy between two maps M_6 and M_5 as a function of the transformation applied to M_5.

4 Multistory FootSLAM

Chapter 3 has introduced entropy-based metrics to measure the uncertainty of FootSLAM maps and the uncertainty regarding a pedestrian's next step. These FootSLAM maps are built on a 2D grid of hexagons and thus do not allow the pedestrian to map different floors of a building in one single walk.

In this chapter we extend FootSLAM to allow mapping and localization in multistory environments. In our work we assume that vertical human motion is relatively rare compared to horizontal motion; we live and work mainly in horizontal floors. *Vertical transitions* (stairways, escalators or elevators) *are thus very informative* and repeated observations of such transitions can act as a strong loop closure event in the FootSLAM estimation process, helping the algorithm converge.

The two main missing pieces that we will address to allow for multistory FootSLAM are the following:

1. Extending the two-dimensional grid of hexagons to allow the mapping of several floors. For this we will use a grid composed of layers of hexagonal prisms (Fig. 4.1).

2. Deriving a suitable error model for the odometry error along the z axis. We will use an autoregressive integrated moving average (ARIMA) model to model the error series of the odometry along the vertical component.

The major challenge of this work is particle depletion: We use a particle filter with a finite number of particles to represent different hypotheses for the state. In a multistory environment the pedestrian is much less likely to revisit regularly areas during his walk ("closing the loop") and the resampling process may eliminate valid hypotheses for the state lying far back in time. In this chapter we will address a number of particle depletion effects and show the resulting FootSLAM maps after processing real-world data collected in three multistory environments. To the knowledge of the author, this has been the first time to show the feasibility of mapping multistory indoor environments using only inertial sensors.

The main contributions of this chapter have been published in [56].

The rest of this chapter is organized as follows: Section 4.1 reviews related work, Section 4.2 presents the theoretical extension of FootSLAM for multistory environments and Section 4.3 addresses particle filter depletion effects. Next, Section 4.4 introduces a model for the vertical component of the odometry error and Section 4.5 describes a method to extract the floor separation from the noisy odometry measurements. To asses the valid-

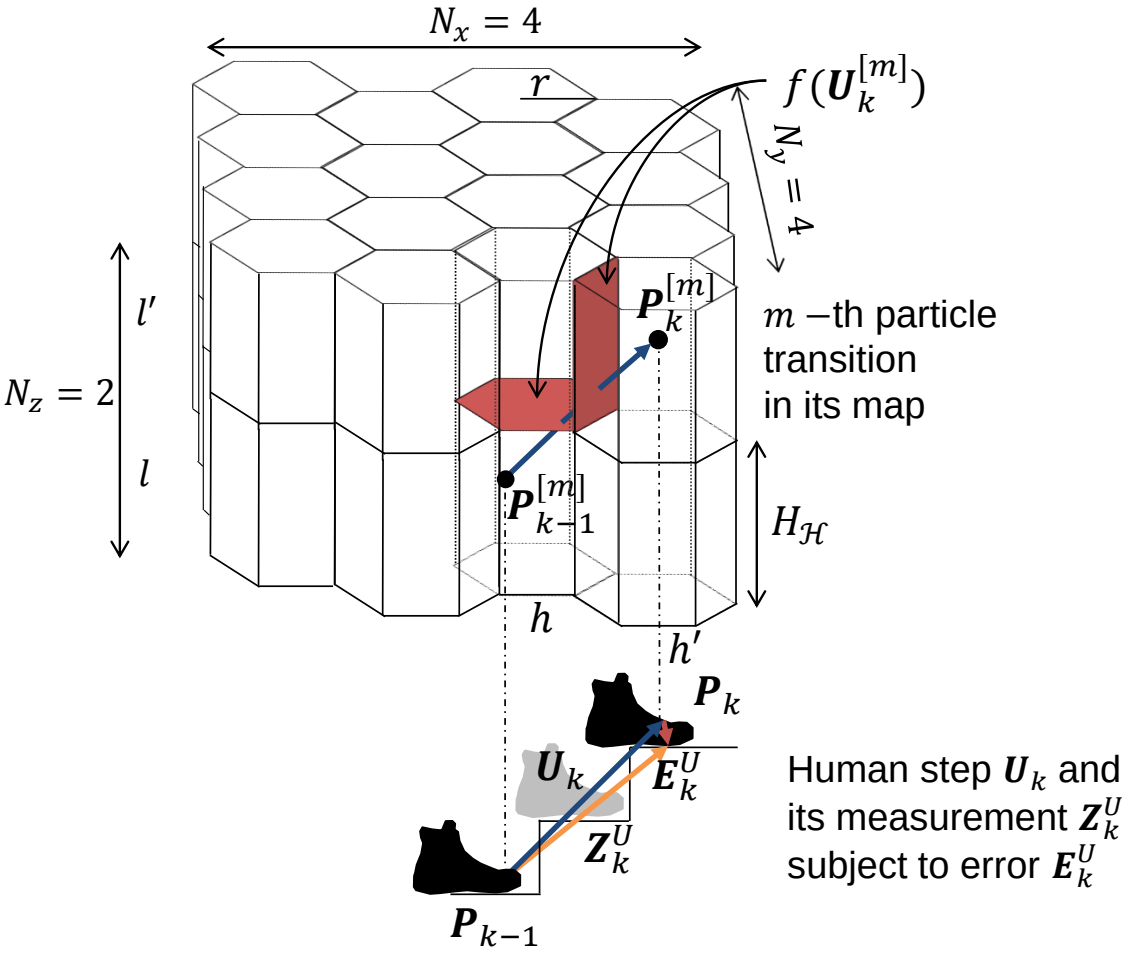


Figure 4.1: On the bottom: The pedestrian takes a step with his right foot (depicted in black) $\boldsymbol{U}_k$, moving from $\boldsymbol{P}_{k-1}$ to $\boldsymbol{P}_k$. We can only observe the measurement of the step $\boldsymbol{Z}_k^U$, which is subject to an additive error $\boldsymbol{E}_k^U$. On the top: The grid of hexagonal prisms discretizes the 3D world. A particle m moves from $\boldsymbol{P}_{k-1}^{[m]}$ to $\boldsymbol{P}_k^{[m]}$, crossing two faces f (marked in red) as a result of its proposed step $\boldsymbol{U}_k^{[m]}$.

ity of the proposed algorithm, Section 4.6 describes three experimental settings and the resulting probabilistic maps.

4.1 Related Work

One problem that arises when mapping multistory buildings is the resulting map misalignment between different floors when no inter-floor constraints are assumed.

The articles [87] and [77] address this problem when mapping the environment with robots: [87] presents an approach that identifies global structural constraints to align the maps of

each floor; [77] uses visual odometry to align the maps of the different floors, which are treated independently.

In the pedestrian navigation community, [142] uses the sensors in a smartphone (accelerometers and gyroscopes) to measure the steps of a pedestrian. The smartphone barometer detects floor changes and map matching is used to revise the position estimates and to remove possible misalignments between the floor maps. Another approach is presented in [48], where the authors address the mapping of a multistory building by collaborating pedestrians (or pedestrians and robots) that wear a vest-mounted array of sensors and whose paths overlap. A barometer detects stairway transitions and an IMU detects elevator transitions. The system generates a series of submaps that are connected when overlapping areas of the map are discovered, but the maps of each floor are treated separately. The extension of actionSLAM to multistory environments [70] relies on the recognition of stair climbing to allow mapping with sub-room accuracy.

4.2 Extension of FootSLAM to Map Multistory Buildings

In FootSLAM, we are interested in estimating the probability distributions of human motion as a function of location. In order to be able to map vertical transitions, the 2D hexagonal grid of Section 2.5 has been now replaced by a grid of uniformly distributed hexagonal prisms.

4.2.1 Hexagonal Prism Grid

For the sake of simplicity we assume that our environment fits within a 3D grid composed of N_x columns, N_y rows and N_z layers of hexagonal prisms (Fig. 4.1). Thus, the number of hexagonal bases is $N_H = N_x \cdot N_y$ and the total number of prisms is $N_{\mathcal{H}} = N_H \cdot N_z = N_x \cdot N_y \cdot N_z$.

A hexagonal prism $\mathcal{H}_i \triangleq \{h, l\} \in \mathcal{H} = \{\mathcal{H}_0, \ldots, \mathcal{H}_{N_{\mathcal{H}}-1}\}$ is a polyhedron with hexagonal base H_h with $0 \leq h \leq N_H - 1$ of radius r, a given height $H_{\mathcal{H}}$ and located at layer L_l with $0 \leq l \leq N_z - 1$ (Fig. 4.1). Thus, the index i or its decomposition into the pair of indices h and l uniquely represent a hexagon's position, with i defined as:

$$i = l \cdot N_H + h.$$

A hexagonal prism has six lateral faces $f \in \{0, \ldots, 5\}$, a lower face $f = 6$ and an upper face $f = 7$.

Later in this chapter we will refer to the coordinates of a prism $\mathcal{H}_i$ as the 3D coordinates $(x, y, z)_{\mathcal{H}_i}$, where $(x, y)_{\mathcal{H}_i} = (x, y)_h$ are the coordinates of the center of its hexagonal base and $z_{\mathcal{H}_i}$ corresponds to the z coordinate at the center of the layer.

4.2.2 Mapping Problem

For the case with a grid of hexagonal prisms (2.50) becomes:

$$p(\boldsymbol{M}|\boldsymbol{P}_{0:k}) = \prod_{i=0}^{N_{\mathcal{H}}-1} p(\boldsymbol{M}_i|\boldsymbol{P}_{0:k}), \tag{4.1}$$

where $\boldsymbol{M}_i = \boldsymbol{M}_{h,l}$ represents the local map of prism $\mathcal{H}_i = \{h, l\}$, that is, an 8-tuple vector of random variables denoting the probability of transition across each face $f \in \{0, \cdots, 7\}$. Thus, the mapping problem is now composed of $N_{\mathcal{H}}$ problems.

Now the transition probabilities of a prism are encoded in:

$$M_{h,l} = (M^0_{h,l}, M^1_{h,l}, M^2_{h,l}, M^3_{h,l}, M^4_{h,l}, M^5_{h,l}, M^6_{h,l}, M^7_{h,l}),$$

and (2.49) becomes:

$$M^{f(\boldsymbol{U}_k)}_{\mathcal{H}(\boldsymbol{P}_{k-1})} = P(\boldsymbol{P}_k \in \mathcal{H}_j | \boldsymbol{P}_{k-1} \in \mathcal{H}_i), \tag{4.2}$$

where $\mathcal{H}_i = \{h, l\}$, $\mathcal{H}_j = \{h', l'\}$ and $\mathcal{H}_i \neq \mathcal{H}_j$ — the pedestrian moved to another prism, $\mathcal{H}(\boldsymbol{P}_{k-1})$ refers to prism where $\boldsymbol{P}_{k-1}$ lies and $f(\boldsymbol{U}_k)$ represents the face of that prism that was crossed when taking step $\boldsymbol{U}_k$ to move to $\boldsymbol{P}_k$ (Fig. 4.1 shows an example with two face crossings). From now on we will use the shortcut i or $\{h, l\}$ for $\mathcal{H}(\boldsymbol{P}_{k-1})$ and f for $f(\boldsymbol{U}_k)$. We can estimate each local map $\boldsymbol{M}_i = \boldsymbol{M}_{h,l}$, the transition probabilities unknown to us, using a vector data structure $C_{h,l}$ whereby each vector element stores the *transition counters* $C^f_{h,l}$ across one of the eight faces.

Finally, the factorized full posterior of (2.51) can be written as:

$$p(\{\boldsymbol{PUE}\}_{0:k}, \boldsymbol{M}|\boldsymbol{Z}^U_{1:k}) = \underbrace{\prod_{i=0}^{N_{\mathcal{H}}-1} p(\boldsymbol{M}_i|\boldsymbol{P}_{0:k})}_{N_{\mathcal{H}} \text{ map estimators}} \cdot \underbrace{p(\{\boldsymbol{PUE}\}_{0:k}|\boldsymbol{Z}^U_{1:k})}_{\text{one pose estimator}}, \tag{4.3}$$

Therefore, the full posterior can be divided into $(N_{\mathcal{H}} + 1)$ estimators, $N_{\mathcal{H}}$ estimators for the computation of the posterior over the map and one for the computation over the pedestrian's poses, step vectors and correlated errors.

4.2.3 Localization Problem

For a grid composed of hexagonal prisms the FootSLAM term (2.61) becomes:

$$I \triangleq \int_{\boldsymbol{M}_{h,l}} p(\{\boldsymbol{PU}\}_k|\boldsymbol{P}_{k-1}, \boldsymbol{M}_{h,l}) \cdot p(\boldsymbol{M}_{h,l}|\boldsymbol{P}_{0:k-1}) \, \mathrm{d}\boldsymbol{M}_{h,l} \propto \frac{C^f_{h,l} + \alpha^f_{h,l}}{C^T_{h,l} + \alpha^T_{h,l}}, \tag{4.4}$$

where $C^f_{h,l}$ is the number of times that face f of prism $\mathcal{H}_i = \{h, l\}$ has been crossed up to time step k and $C^T_{h,l} = \sum_{f=0}^{f=7} C^f_{h,l}$ and $\alpha^T_{h,l} = \sum_{f=0}^{f=7} \alpha^f_{h,l}$. Thus, particles that revisit similar transitions obtain a greater reward over time.

The term $\alpha^f_{h,l}$ refers to the virtual prior counts, which in [128] had the empirically chosen value of 0.8 for all six lateral edges (now lateral faces $f \in \{0, \ldots, 5\}$) (see Section 2.5). In Section 4.3.1 we address the use of the virtual prior counts for the lateral faces in more detail. On the other hand, given that vertical transitions such as stairways, ramps, escalators and elevators are relatively rare, we can assume that the prior probability of human vertical motion is lower than the prior probability of human horizontal motion. For this reason, we have empirically chosen $\alpha^f_{h,l} = 0.05\,,\ \forall\ f \in \{6, 7\}\ \forall\ h, l$. Thus, when the pedestrian revisits a vertical transition, it helps the convergence of the map, acting as a wormhole in the particle filter estimation process.

4.2.4 Changes to the RBPF algorithm for Multistory FootSLAM

The RBPF algorithm used in FootSLAM is detailed in Section 2.5 and in [8]. The two main steps of the algorithm are the proposal and the weight update steps, which we explain next.

Proposal Step

As mentioned in Section 2.5.7, Algorithm 3 in [134] describes the proposal step for the horizontal odometry error component and consists of a slow and fast angular random walk process and a white noise component. The particular proposal density function for the vertical component of the odometry errors is studied in Section 4.4.

Weight Update Step

The weight update step takes into account the product of all instances of (4.4) for each face crossed. In [128] it had been proposed to apply a non-linear function to the weight update based on the angle of the current step vector in order to account for directional biases in the hexagonal grid. In this work we have adopted a weight computation that accounts for steps spanning many faces and which does not suffer from directional biases. The underlying thought is to see a weight update as a *reward for a particle having traveled in a particular direction*; the hexagonal grid leads to a relatively coarse angular and spatial discretization. In order to make the weight update as invariant to the hexagonal grid as possible (including the hexagonal prisms' orientation in space, radius and height) we will compute the geometric mean of the weight update (4.4) and account for the accrued change in traveled odometry O_c. The use of the geometric mean is motivated by the fact

that weights are multiplicative. Thus, (2.67) becomes:

$$w_k^{[m]} \propto w_{k-1}^{[m]} \cdot \left\{ \prod_{j=1}^{N_c^{[m]}} I_j^{[m]} \right\}^{\eta^{[m]}}, \tag{4.5}$$

where $N_c^{[m]}$ is the number of faces crossed by particle m since the last weight update and $I_j^{[m]}$ is computed using (4.4) for the j-th face crossing of particle m (for example in Fig. 4.1, $N_c = 2$). We define $\eta^{[m]}$ as follows:

$$\eta^{[m]} = \frac{\varsigma \cdot O_c}{N_c^{[m]}}, \tag{4.6}$$

where O_c is the accrued change in traveled odometry and ς is a scale factor: We are incorporating one "face's worth" of (4.4) per $\varsigma = 1.1$ meters of traveled odometry.

We have observed sudden stark weight contrasts that sometimes immediately trigger resampling, which results in a pronounced loss of diversity. Therefore, following Algorithm 4 in [134], we retain temporary weights for each particle that are only multiplied into the existing weight after having been geometrically averaged using (4.5) over a sufficiently large odometry (here we wait six times the radius of the hexagons).

4.3 Effects of Depletion

A problem encountered of using particle filters with a finite number of particles is the loss of diversity of hypotheses describing the state space lying far back in time [13]. This is due to the resampling step whereby locally good particles tend to be duplicated and weak particles are often erased. This causes particle depletion in areas that had been visited in the past, since the resampling process eliminates hypotheses that might have been valid in the previous time points. In the next subsections we address two problems that arise from this depletion effect and that represent the two main challenges of multistory FootSLAM.

4.3.1 Floor Discontinuity Problem

In the two dimensional world we already experience very large loops but for the experimental data [128][57][136][61], we have nonetheless achieved convergence. However, in a large multistory building, the length of the loops can be much bigger because of the additional freedom to explore more stories; this appears to be compounded in buildings with multiple stairways.

In particular, when a pedestrian visits a given floor intermittently and the time between visits is relatively long, depletion in the number of hypotheses for the floor height can cause a *discontinuity* in the map: due to the drift and other errors of the odometry the

different visits to the floor may be mapped into multiple prism layers, each with a different elevation.

To tackle this problem, we assume that the floor separation S_{floor} is constant and that it is known or can be inferred from the odometry data (as explained in Section 4.5). Given the floor separation, we can compute which prism layers coincide with a floor. Since the presence of horizontal walkable areas in between floors is relatively scarce, it is reasonable to assume that the prior likelihood of horizontal motion within prism layers that coincide with a floor is higher than that likelihood within prisms layers that do not coincide with a floor (for example during stairway transitions). To avoid the discontinuity problem we propose to set slightly stronger virtual prior counts at those prism layers L_l that coincide (symbol $\leftrightarrow$) with a floor:

$$\begin{aligned} \alpha^f_{h,l} &= 0.8,\ f \in 0,\ldots,5,\ \forall\ h && \forall\ l \leftrightarrow \text{floor} \\ \alpha^f_{h,l} &= 0.5,\ f \in 0,\ldots,5,\ \forall\ h && \forall\ l \nleftrightarrow \text{floor}. \end{aligned} \tag{4.7}$$

These values have been empirically chosen after careful visualization of the particle cloud during the FootSLAM estimation process for many datasets.

4.3.2 Misalignment Between Floor Maps

FootSLAM is rotationally invariant if we do not use any heading sensor. Ideally, FootSLAM's posterior distribution would retain a high degree of angular diversity (relative to any initial heading that is assumed at $k = 0$). However, due to the finite number of particles, typically a few angular hypotheses will remain after some time. When a pedestrian has spent some time on a floor, the resulting small set of map hypotheses for that floor might not necessary overlap with the maps on other floors (see Fig. 4.2).

However, we can assume a certain degree of structural similarities between the different floors in the building, namely the position of stairways, lifts or main corridors. We can include a prior factor $g(\cdot)$ in the FootSLAM term (2.70) that takes into account the number of transitions across the crossed face f at the other prism layers with the same hexagonal base H_h. We obtain:

$$I = \frac{C^f_{h,l} + \alpha^f_{h,l} + g(K^f_h - C^f_{h,l})}{C^T_{h,l} + \alpha^T_{h,l} + g(K^T_h - C^T_{h,l})}, \tag{4.8}$$

where $K^f_h = \sum_{l=0}^{l=N_z-1} C^f_{h,l}$, is the aggregated transition counter corresponding to the crossed face f for all layers in the grid with hexagonal base H_h, and $K^T_h = \sum_{f=0}^{f=7} K^f_h$.

In our experiments we use $g(K^f_h - C^f_{h,l}) = \beta \cdot (K^f_h - C^f_{h,l})$ with degree of structural similarities $\beta = 0.3$ and for quick computation of the weight update equation, we also store the aggregated counters K^f_h during the map update step. Other non-linear functions $g(\cdot)$ are to be studied in future work.

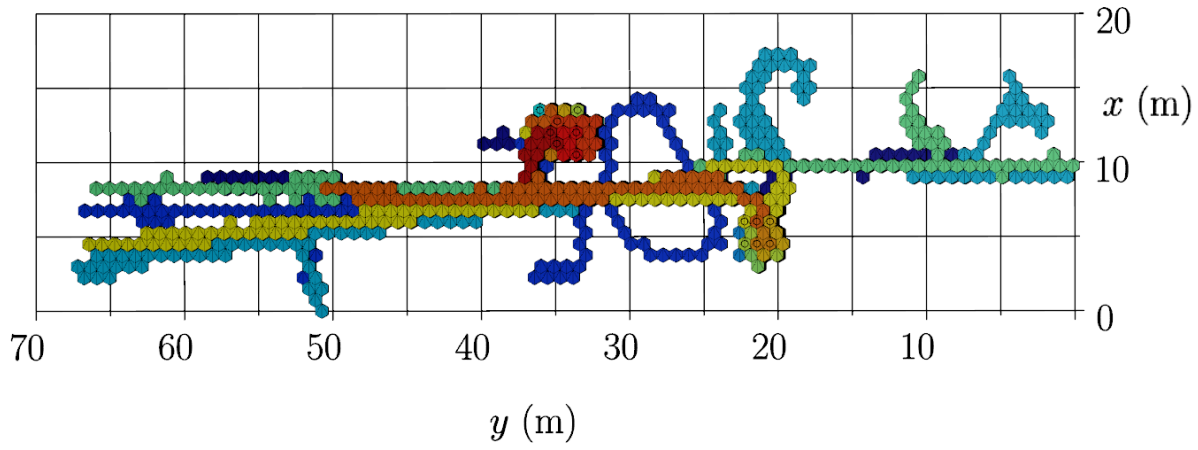


Figure 4.2: Illustration of a moderate presentation of the misalignment effect: the top view of a FootSLAM map corresponding to Walk 1 (see Section 4.6 for more details). Different floors are marked with different colors. There is a clear misalignment between the corridors of the different floors.

Proceeding in this way, we can correct possible misalignments between different floors in an online fashion (Fig. 4.3), as opposed to the approaches in [87] and [142], which correct the floor misalignment in a post-processing step.

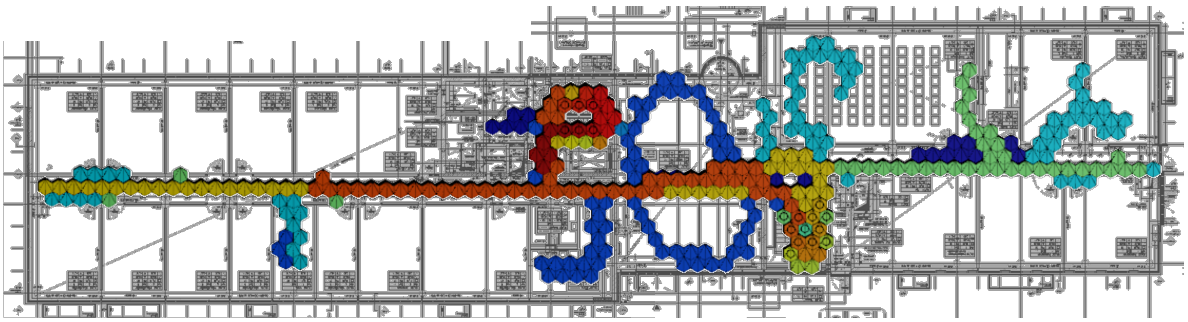


Figure 4.3: Top view of a FootSLAM map corresponding to Walk 1 (see Section 4.6 for more details). The misalignment has been corrected using the aggregated counts of the other levels as inter-floor constraint. The floor plan of the ground floor is shown for comparison, which is why hexagons in colors other than dark blue may appear to violate some of the walls that are specific to the ground floor such as the one separating the two upper right rooms.

4.4 Modeling the Stochastic Drift of the z Component of the Odometry

As mentioned in Chapter 2, FootSLAM splits the estimation process into two levels: The UKF in [166] integrates the noisy IMU measurements and an upper level RBPF processes the UKF odometry output. The particles in the RBPF propose errors (first term in (2.66)) in order to model the drift in the odometry. As mentioned in Chapter 2, FootSLAM splits the estimation process into two levels: The UKF in [166] integrates the noisy IMU measurements and an upper level RBPF processes the UKF odometry output. The particles in the RBPF randomly propose errors (first term in (2.66)) in order to model the drift in the odometry.

The main error sources that distort the 2D odometry used in this section are introduced in Section 2.5.7. In this contribution, we focus on finding a suitable model for the errors that perturb the z component of the odometry, visible in Figures 4.4 and 4.9. For a complete review of the error sources in INSs, the reader is referred to [165].

4.4.1 Computing the Error

For simplicity, we will study the error present in the vertical component of the odometry resulting from a walk on a single floor (Fig. 4.4). We assume a constant floor height and thus any change from the initial z measurement corresponds to errors.

ARIMA models have been widely used to model many empirical signals such as electricity prices, stock prices and storms, when random changes in the slope of the series — trends (in the example a negative trend)— are present [23]. Examples of ARIMA models within the field of inertial navigation can be found in [115] and [62], where the drift of gyroscopes is modeled accordingly. However, our goal is to model the *odometry*, not the error present in the raw measurements collected with inertial sensors.

4.4.2 ARIMA Model

An ARIMA model is composed of an autoregressive (AR) model of order p, d integrations (I) and a moving average (MA) model of order q. The ARIMA(p, d, q) model can be expressed as follows [23]:

$$\phi(B)\nabla^d z_k = \theta(B)w_k, \tag{4.9}$$

where

- z_k represents the series we wish to model and w_k are independent i.i.d samples from a normal distribution $\mathcal{N}(w; 0, \sigma_w)$.
- B is the backward shift operator, i.e. $B^m \cdot z_k = z_{k-m}$.
- $\nabla = (1 - B)$ is the backward difference operator, i.e. $\nabla z_k = z_k - z_{k-1}$.

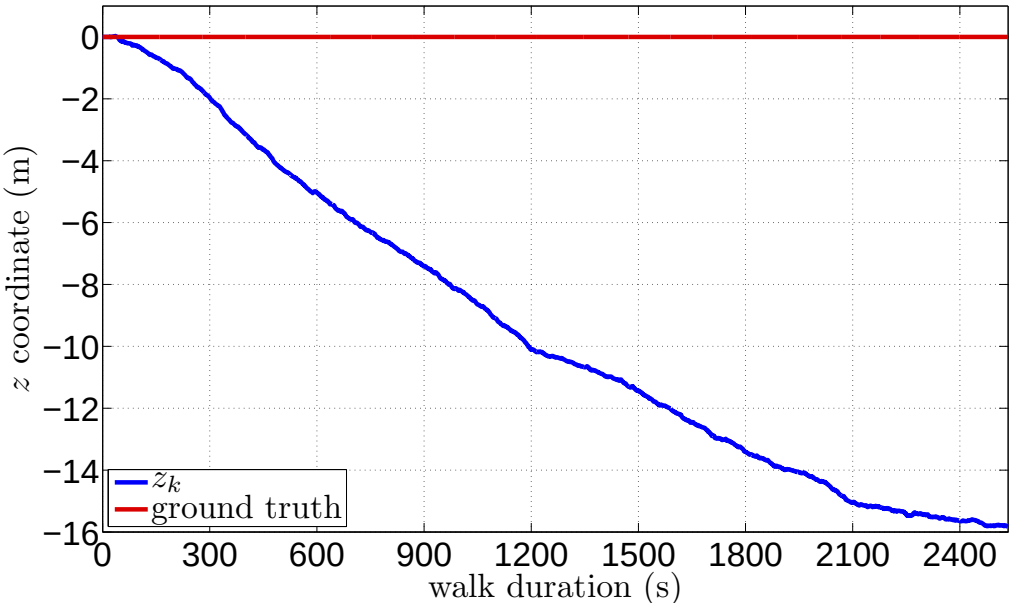


Figure 4.4: Odometry along the vertical axis, z_k, collected during one walk within one floor (in blue) with respect to the ground truth (in red). In this case the odometry corresponds to the error present in the series.

- $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 \cdots - \phi_p B^p$ is a stationary AR operator (its roots lie outside the unit circle) of order p.
- $\phi(B)\nabla^d$ is the generalized (non-stationary) AR operator with d roots equal to unity.
- $\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \theta_q B^q$ is an invertible MA operator (its roots lie outside the unit circle) of order q.

Two steps help identify an ARIMA model [23]:

1. Iteratively difference z_k until the resulting series $y_k^{(d)} = \nabla^d z_k$ becomes stationary.
2. Identify the resulting autoregressive-moving average (ARMA) series, ARMA(p, q):

$$\phi(B) y_k^{(d)} = \theta(B) w_k.$$

We can rely on the sample Autocorrelation Function (ACF) (see Appendix A.1) and sample Partial Autocorrelation Function (PACF) (Appendix A.2) to estimate the model orders p, d, q.

Note that we will follow these steps to model the error that perturbs the vertical component of the odometry obtained using the UKF algorithm in [166]; the model orders and parameters will only apply to odometry obtained with that algorithm. Nevertheless, the mathematical procedure is general and remains valid for any data with a stochastic drift.

4.4.3 Proposed Model

For clarity, we first introduce the error model and postpone its validation from data. Our hypothesis is that the error series z_k can be represented by an ARIMA(0,2,2) model. We define y_k as the one time differenced series $y_k = \nabla z_k$ and x_k as the two-time differenced series $x_k = \nabla^2 z_k$. The series $\tilde{y}_k$ refers to y_k with the mean μ_y removed, Δt_k to the time elapsed between step k and step $k-1$ and d_k to the drift series. n_k is the driving noise series of the drift, generated by a MA(2) filter with parameters θ_1 and θ_2 that filters independent i.i.d. samples from a normal distribution $w_k \sim \mathcal{N}(w; 0, \sigma_w)$ (Fig. 4.5).

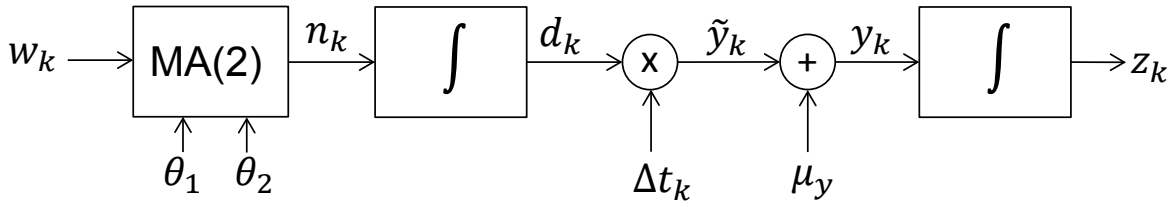


Figure 4.5: Diagram of the error model for the vertical component of the odometry. w_k are independent i.i.d. samples from a normal distribution $\mathcal{N}(w; 0, \sigma_w)$. MA(2) stands for moving average filter of order 2.

The series are related as follows:

$$\begin{aligned} z_k &= z_{k-1} + y_k && I(1) \\ y_k &= \tilde{y}_k + \mu_y \\ \tilde{y}_k &= \Delta t_k \cdot d_k \\ d_k &= d_{k-1} + n_k && I(1) \\ n_k &= w_k - \theta_1 w_{k-1} - \theta_2 w_{k-2} && MA(2). \end{aligned} \tag{4.10}$$

For simplicity, we assume that the series is evenly spaced with Δt_k =1 s, $\forall\, k$. Differencing the error series z_k yields:

$$y_k = \nabla z_k = \tilde{y}_k + \mu_y = d_k + \mu_y, \tag{4.11}$$

which is a non-stationary series, given the non-stationarity of d_k. Differencing the series one more time results in:

$$x_k = \nabla^2 z_k = \overbrace{d_k - d_{k-1}}^{n_k} = w_k - \theta_1 w_{k-1} - \theta_2 w_{k-2}, \tag{4.12}$$

which is stationary for all values of θ_1 and θ_2 [23]. Hence, differencing two times ($d = 2$) renders the original z_k series stationary, as expected for a model with two integrations.

Given that the autocovariance (A.1) of w_k is $c_{ww}(l) = \sigma_w^2\ \delta(l)$, the autocovariance of the two-time differenced series x_k is obtained by simple insertion of (4.12) in (A.1):

$$c_{xx}(l) = \sum_{m=-2}^{m=2} \gamma_m\ \delta(l-m), \tag{4.13}$$

with autocovariance coefficients:

$$\gamma_0 = \sigma_w^2(1+\theta_1^2+\theta_2^2) \tag{4.14}$$
$$\gamma_{-1} = \gamma_1 = -\sigma_w^2\theta_1(1-\theta_2) \tag{4.15}$$
$$\gamma_{-2} = \gamma_2 = -\sigma_w^2\theta_2, \tag{4.16}$$

which is the well-known result for a moving average model of order 2 [23].

4.4.4 Model Identification

In this section we identify the model orders for the error series z_k depicted in Fig. 4.4. The sample ACF and the sample PACF of z_k, not shown here for brevity, correspond to a non-stationary series, since the ACF does not tend to zero and the PACF is negligible from lag 2 [23].

Differencing the series z_k as in (4.11) yields $\mu_y = -0.00623$ m, an offset that causes the negative trend. The ACF and PACF of $\tilde{y}_k$ decay quickly (Fig. 4.6) but die rather slowly — we have plotted two lines that correspond to the 95% confidence in the estimates of the ACF and PACF, indicating when the coefficients are negligible (see Appendices).

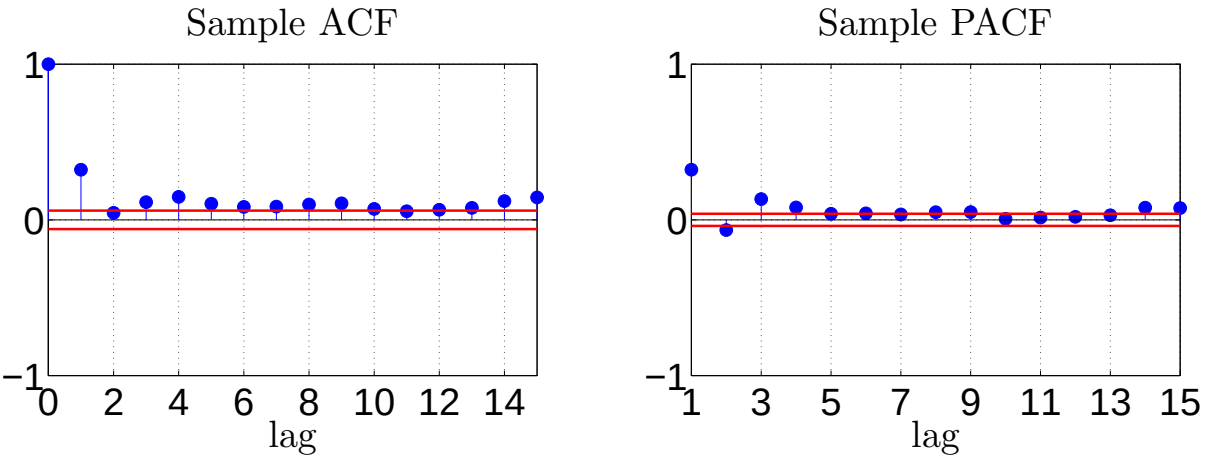


Figure 4.6: Sample ACF and sample PACF of $\tilde{y}_k$. The limits for the 95 % confidence interval are shown as a reference (see Appendices).

A second differentiation (4.12) results in a fast decaying ACF (with negligible coefficients from lag 3) and a PACF that decays quickly but dies slowly (Fig. 4.7). These plots suggest that x_k is stationary [23]. Thus, the number of integrations d is 2.

The next step is to identify the resulting ARMA series. Since the PACF dies slowly and the ACF is negligible from lag 3, a MA(2) seems suitable to model the series as shown in (4.13) (see also Appendices). As a result, our proposed model follows an ARIMA(0,2,2).

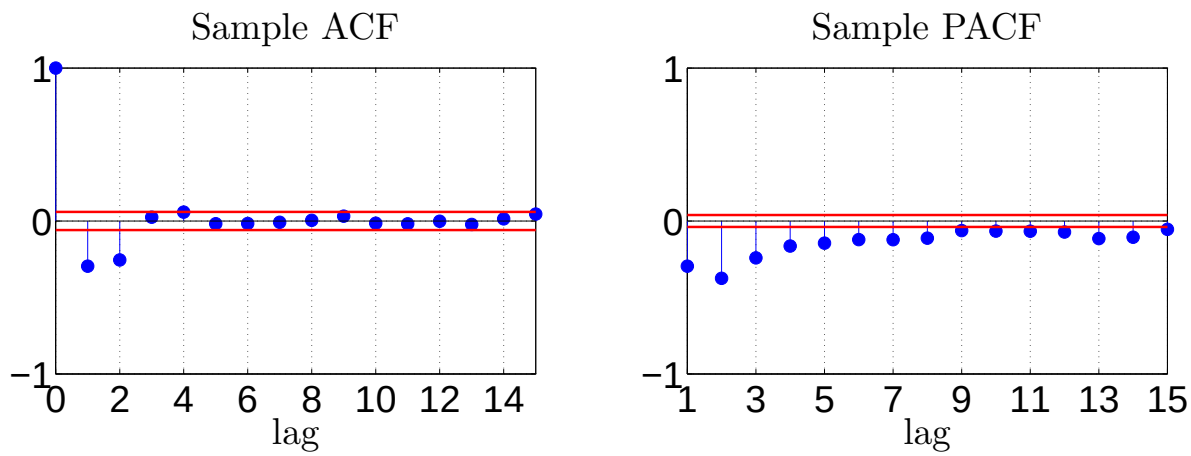


Figure 4.7: Sample ACF and sample PACF of x_k. The limits for the 95 % confidence interval are shown as a reference (see Appendices).

4.4.5 Parameter Estimation of MA(2)

Computing the autocovariance (A.1) of our example series x_k and identifying the coefficients γ_0 (4.14), γ_1 (4.15) and γ_2 (4.16) yields $\theta_1 = 0.636$, $\theta_2 = 0.354$ and $\sigma_w = 8.85 \cdot 10^{-3}$ m/s.

Fig. 4.8 shows the periodogram estimate [119] of the spectrum of x_k (in blue) and the theoretical spectrum of x_k with the estimated parameters (in red). In our view, this figure supports the validity of our model.

We have applied the model individually to 10 walks with a total duration of ca. 4 hours. Table 4.1 shows the mean and standard deviation of the model parameters, which are consistent for different walks. Furthermore, the offset μ_y is on average -8.47 mm; however in our implementation we have set it to 0.0 m.

	θ_1	θ_2	σ_w
mean	0.614	0.305	$1.28 \cdot 10^{-2}$ m/s
std. dev.	0.0895	0.0566	$3.34 \cdot 10^{-3}$ m/s

Table 4.1: Mean and standard deviation of the parameters of the MA(2) model for the two-time differenced series x_k.

The proposal density function of the RBPF consists of the normal distribution $w_k \sim \mathcal{N}(w; 0, \sigma_w)$ and the error series will be generated as in Fig. 4.5. The recursion starts with $w_{-2} = w_{-1} = 0.0$ and we draw w_0 from $\mathcal{N}(w; 0, \sigma_w)$ and d_{-1} from $\mathcal{N}(d; 0, \sigma_d)$ with $\sigma_d = 0.05$ m/s (chosen empirically).

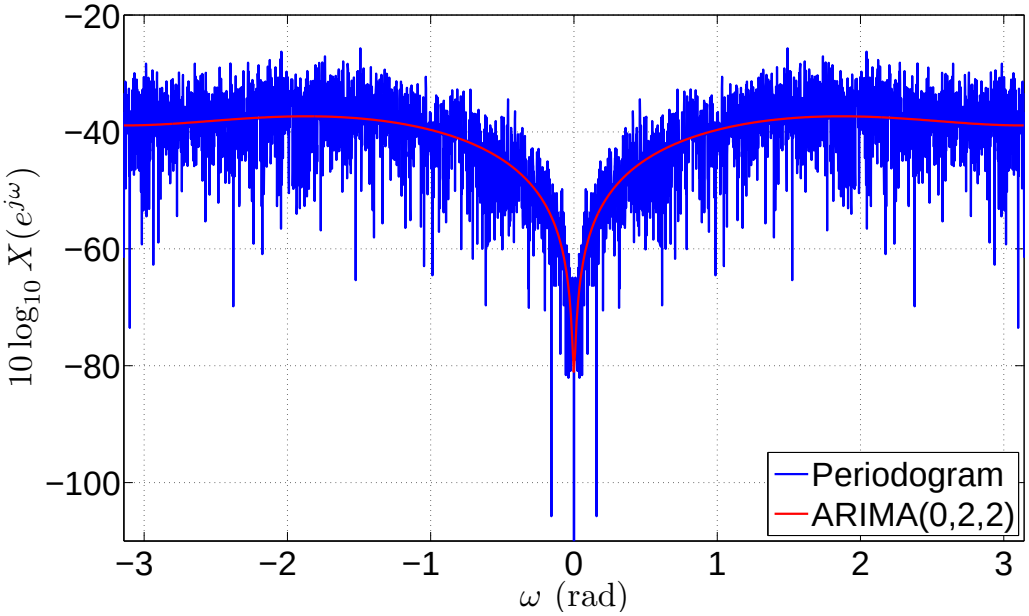


Figure 4.8: Periodogram spectrum estimate $X(e^{j\omega})$ of the two-time differenced series $x_k = \nabla^2 z_k$ with z_k as shown in Fig. 4.4 (blue curve) and theoretical spectrum estimate for the proposed model in (4.10) (red curve).

4.5 Estimating the Floor Separation

In this work, we extract the floor separation from the odometry data under the following assumptions:

- The separation between the floors in the building is constant.
- The pedestrian does not use elevators or escalators, but only staircases.

Given the odometry data such as those in Fig. 4.9 (in blue), the steps to estimate the floor separation are:

1. Detect stairway transitions.
2. Compute the vertical lengths of the stairway transitions, L_{stw}. Note that the stairway transitions can correspond to more than one floor.
3. Group the lengths into B bins with mean μ_j, $\mathcal{B} = \{B_1, B_2, B_j, \ldots, B_B\}$.
4. Obtain an estimate of the floor separation $\hat{S}_{\text{floor}}$ in the building given the bins.

Steps 1 to 3 are performed using Algorithm 1 which is based on Algorithm 1 in [48] but uses bins instead of a Gaussian Mixture Model and odometry data instead of altimeter data. Step 1 is achieved by using the differenced odometry $y_k = \nabla z_k = z_k - z_{k-1}$ and computing its standard deviation std$(\cdot)$ within a window W_k. During stairway transitions, the standard deviation is greater than in floor areas. The threshold t_{std} has been set to

0.1 m. The method add(·) appends the last computed stairway length to the bins $\mathcal{B}$ by choosing the bin with the closest mean or adding a new bin when the distance to any mean is greater than a second threshold $t_{\text{bin}} = 0.5$ m.

Algorithm 1 StairwaysDetection

Input: z_k, y_k
Output: $\mathcal{B}$

```
onFloor =true
index = 1
for each time step k do
    isFloor_k = std(y_k · W_k) < t_std // W_k is a rectangular sliding window of length L centered at k
    if isFloor_k and not onFloor then
        onFloor =true // walking out of the stairway
        f = max(k − L/2, 1) // finishing index of the stairway
        L_stw = z_f − z_s // L_stw is stairway length
        add(B, L_stw)
    else if not isFloor_k and onFloor then
        onFloor =false // walking into the stairway
        s = min(length(y_k), k + L/2) // starting index of the stairway
    end if
end for
```

Given the set of bins $\mathcal{B}$ that contain the stairway transition lengths we can estimate the floor separation by finding the (possibly) missing fundamental L_0, i.e. the Greatest Common Divisor (GCD) of the B non-integer numbers μ_j, $\forall\, j \in \{1, \ldots, B\}$. Thus, $\hat{S}_{\text{floor}} = L_0$. In our example of Fig. 4.9, we obtain three bins about $\mu_1 = 3.667$ m, $\mu_2 = 7.183$ m, $\mu_3 = 14.273$ m — corresponding to 1, 2 and 4 floor transitions, respectively. The estimated floor separation results in $\hat{S}_{\text{floor}}$ =3.591 m and the real floor separation is S_{floor} =3.54 m. Thus, the error for this dataset is of the order of 10^{-2} m.

We can now compute the elevation of each floor with respect to the ground floor and the corresponding floor numbers, as shown in Fig. 4.9 (in green). Note that we can freely choose the number of intermediate prism layers between two floors (FootSLAM maps vertical resolution), but typically we use 3 to 10 layers. We set stronger horizontal priors on those layers that coincide with a floor, applying (4.7).

4.6 Experimental Verification and Results

In this section we evaluate the performance of the proposed multistory FootSLAM algorithm in three multistory environments by presenting the resulting maps.

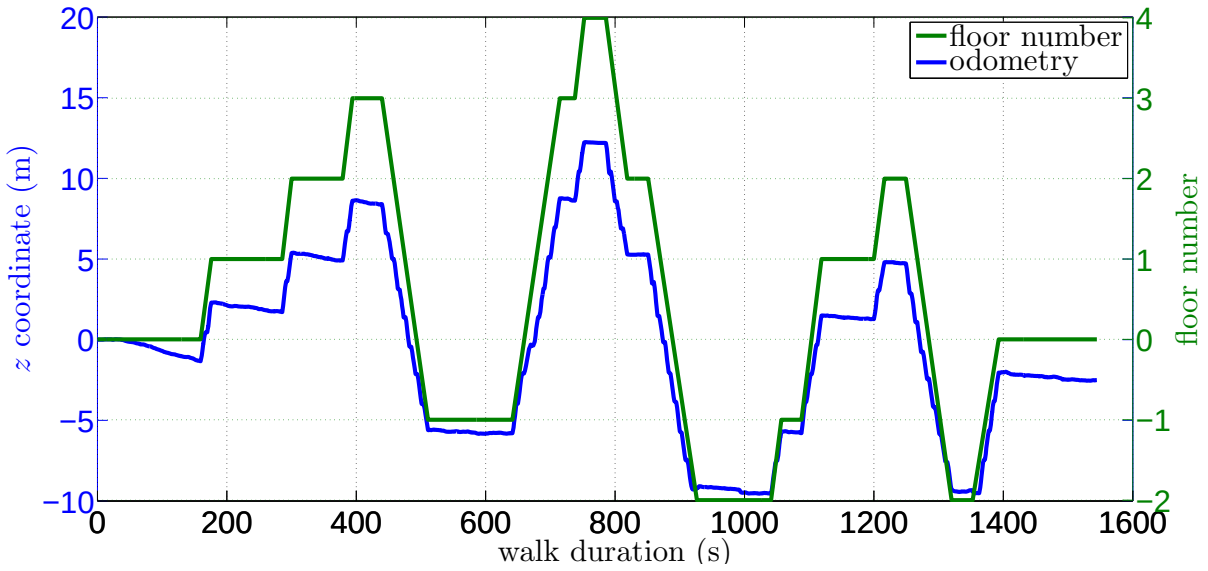


Figure 4.9: Odometry along the z axis for one walk (in blue). A negative drifting error is clearly visible. In green, the floor number (starting from 0) have been computed following the algorithm explained in Section 4.5.

Walk	S_{floor}	$\hat{S}_{\text{floor}}$	$H_{\mathcal{H}}$	N_z	Location	Floor Area	N_{stw}	N_{floor}	Duration
1	3.54m	3.591m	1.18 m	32	DLR building 103	70 m×20 m	2	7	26 min
2	3.49m	3.503m	0.5817 m	64	TUM building N4	25 m×25 m	2	7	18 min
3	3.58m	3.522m	0.597 m	64	TUM building 9	15 m×110 m	3	5	36 min

Table 4.2: Particular settings for each walk.

4.6.1 Experiment Settings

A pedestrian walked with a shoe-mounted IMU, the cable-connected MTx - Xsens®, that measured his steps while walking. The data were recorded in August and September 2011. The UKF algorithm described in [166] was used as the dead reckoning algorithm. We deployed $N_p = 30\,000$ particles for the RBPF and the following parameters for the predefined prism grid: $r = 0.5$m, $N_x = 256$, $N_y = 256$.

The pedestrian undertook three walks within three different buildings: Walk 1, Walk 2 and Walk 3 (Figs. 4.10, 4.11 and 4.12, respectively). Walk 1 was undertaken at building 103 at DLR whereas Walk 2 and Walk 3 took place at TUM buildings 9 and N4, respectively. Table 4.2 shows the particular settings for the prism height $H_{\mathcal{H}}$ and the number of prism layers N_z along with estimated floor separation $\hat{S}_{\text{floor}}$ of each building. The true floor separation S_{floor} was known. In addition, the table shows the floor area of one floor (roughly), the number of stairways used N_{stw}, the number of visited floors N_{floor} and the duration of each walk.

During the three walks, the pedestrian walked within the floors and from one floor to the next using the stairway, i.e. no elevators or escalators were used. The pedestrian walked at a normal pace and during office hours, avoiding obstacles such as other walking pedestrians and furniture.

4.6.2 Results

Figures 4.10(c), 4.11(b) and 4.12(b) show the resulting *maximum a posteriori* FootSLAM maps for the three walks. Different colors represent different floors and blue rectangular polyhedrons correspond to non-floor prism layers, indicating ramps or stairway transitions. For comparison we have also depicted the odometry measurements of Walk 1 (Fig. 4.10(b)), which clearly suffers from heading drift. Interactive PDF files with the FootSLAM maps are available at [55].

Walk 1

Walk 1 took place at German Aerospace Center (DLR) office building 103 (Fig. 4.10(a)). The building covers an area of roughly 70 m×20 m. The walk had a duration of ca. 26 min, covering 6 floors and the roof. Two sets of stairways were visited both going up and down. Figure 4.10(c) shows the resulting FootSLAM map.

Walk 2

Walk 2 took place at Technical University of Munich (TUM) office building N4 (Fig. 4.11(a)). The building covers an area of roughly 25 m×25 m. The walk had a duration of ca. 18 min, covering a total 7 floors also using two stairways. Figure 4.11(b) shows the resulting FootSLAM map.

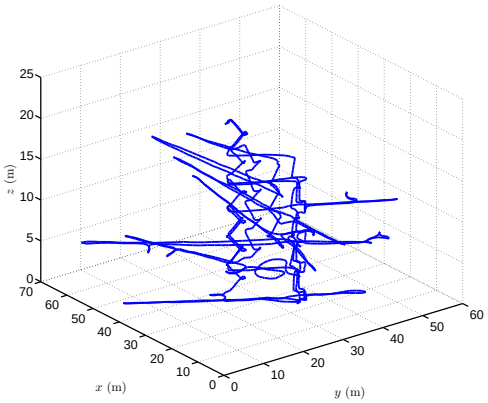
Walk 3

Walk 3 took place at TUM office building 9 (Fig. 4.12(a)). The building covers an area of roughly 15 m×110 m. The walk had a duration of ca. 36 min, covering a total 5 floors. Three sets of stairways and a ramp were used during the walk. Figure (Fig. 4.12(b)) depicts the resulting the FootSLAM map. The pedestrian walked outdoors for a while, corresponding to the wide loop.

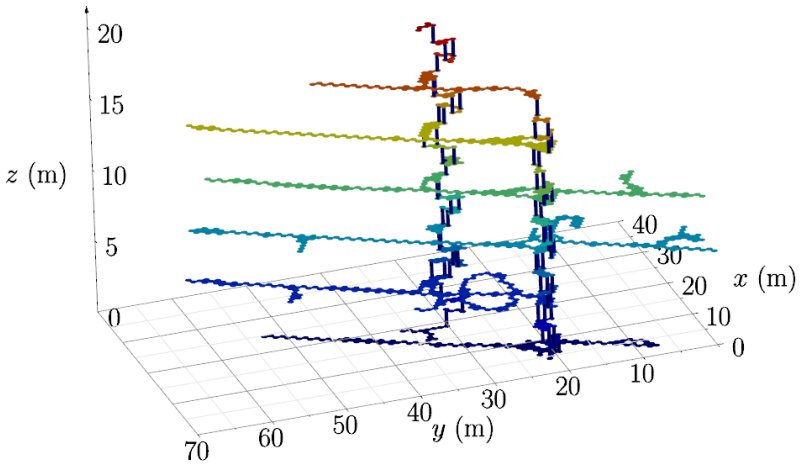
4.7 Discussion

We have shown that FootSLAM is a suitable technique to estimate the probabilistic maps of multistory environments using data collected using inertial sensors. Contrary to what

(a) DLR building 103.



(b) Odometry.

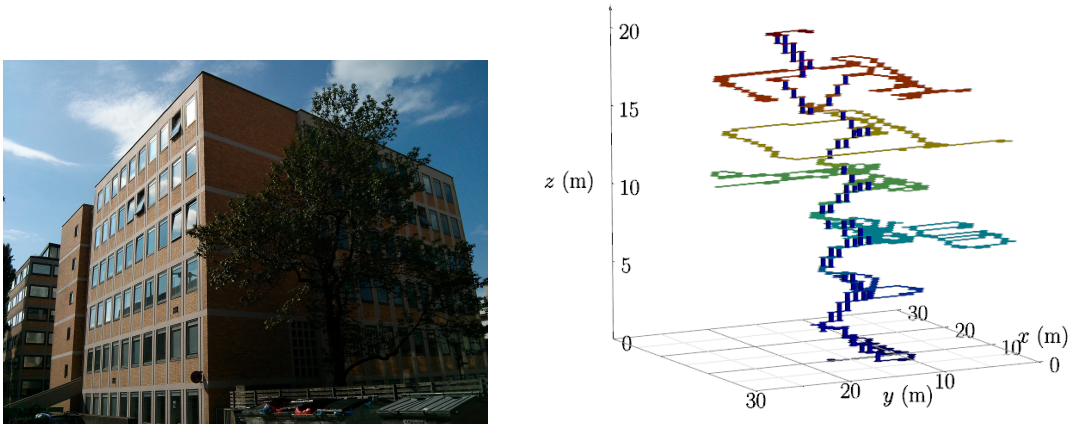


(c) FootSLAM map.

Figure 4.10: Building where the Walk 1 took place (Fig. 4.10(a)), its corresponding odometry (Fig. 4.10(b)) and the resulting FootSLAM map (Fig. 4.10(c)).

one might expect, the problem of multistory mapping is much more difficult than mapping 2D areas. This is because of the extra freedom of the pedestrian to explore much larger loops and the relatively small inter-story separation.

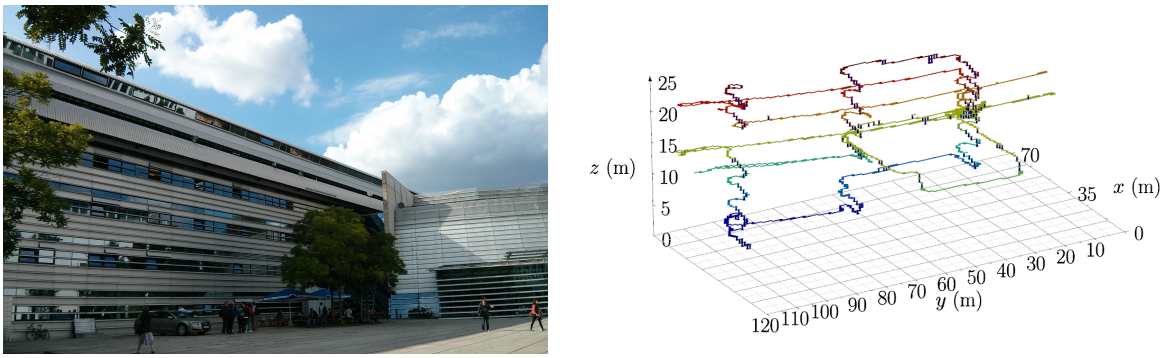
The main challenges that we have encountered in this work have been 1) The drifting odometry, for which we have derived a suitable model representation, and 2) The two problems associated to particle depletion: the discontinuity problem and the misalignment problem. To address these problems, we rely on the assumption that artificial human environments are more likely to exhibit horizontal walkable surfaces over vertical transitions. Then, we have addressed the floor discontinuity problem by measuring the floor separation in the building or estimating it from the collected data and by setting



(a) TUM building N4.

(b) FootSLAM map.

Figure 4.11: Building where the Walk 2 took place (Fig. 4.11(a)) and the resulting FootSLAM map (Fig. 4.11(b)).



(a) TUM building 9.

(b) FootSLAM map

Figure 4.12: Building where the Walk 3 took place (Fig. 4.12(a)) and the resulting FootSLAM map (Fig. 4.12(b)). In this case three stairways were used and a ramp outdoors.

stronger priors for prism layers that coincide with a floor; we have addressed the misalignment between floor maps by assigning a degree of structural similarities between the floors $\beta = 0.3$ that still allows the mapping of story-specific areas and wide corridors. Note that in these three examples there are strong structural similarities such as the position of the main corridors and stairways and further work will need to include other experiments in which there are no such symmetries across the floors.

Besides the results presented for three large buildings, we have conducted numerous other experiments (not shown for brevity and including other buildings) with similar results. Due to the scale of the experiments, we could not precisely track the true path followed by the pedestrian during the walks, making it impossible to compute the error in the positioning estimate over time. However, we have been able to verify for all experiments

that the visits of the pedestrian to each floor were mapped onto the corresponding prism layer. In addition, we have manually aligned the FootSLAM map of each floor with the available building floor-plan (as in Fig. 4.3) and we have confirmed by visual inspection that each floor map reflects the visited offices (see [22]).

The accuracy of the maps — meaning its ability to reflect doors, pathways, obstacles, etc. — depends on the chosen spatial resolution, namely the radius of the hexagons r and the height of the prisms $H_{\mathcal{H}}$. The chosen settings, $r = 0.5$ m and $H_{\mathcal{H}}$ as shown in Table 4.2 represent a good compromise between resolution and memory requirements. In [134] we explore the use of a hierarchical grid with different radii to map the local disturbances of the magnetic field; such an approach could be useful for FootSLAM but care must be taken since using too small or too big cells would violate the local map independence assumptions of FootSLAM [8].

The main practical limitation of the work presented here is that our validation of multistory FootSLAM is based on a foot-mounted IMU. The next milestone towards mass-market applications is to replicate these results using the sensors of a smartphone even when the user is performing different activities such as wandering, running, jumping, texting, calling, etc. [151].

Future work shall address the mapping of much larger buildings, structures with different floor separations and weak structural similarities across floors. Hereby the use of other sensors such as altimeters, magnetometers, WiFi signal strength and GPS is expected to ensure the robustness of FootSLAM. For example, we have processed the data from Walk 1 successfully applying magnetic field intensity SLAM [134]; the results are discussed in [133]. Two videos that show the particle filter estimation process in two different buildings are available online [132][135].

5 Reducing the Complexity of FootSLAM

Chapter 4 has extended FootSLAM to allow mapping multistory buildings. However, as discussed in Section 2.5.5, the data structure used to store the particles maps during the FootSLAM estimation process do not allow real time processing of the odometry measurements collected when exploring large areas.

In this chapter we analyze the complexity of the existing FootSLAM algorithm implementation (Section 5.1), identify main bottlenecks and present a new geographic tree-based data structure, called H-tree (Section 5.2), that reduces the computational complexity of FootSLAM and allows quasi real-time processing. Building on the H-tree, we also propose an approach to size-adaptive FootSLAM (Section 5.3), whereby the mapping grid can grow as new areas are explored.

Additionally, in this chapter we introduce an alphabet of symbols to compactly represent the eight transition counters that store the number of times that each particle has crossed the faces of a hexagonal prism (Section 5.4). This will result in memory savings and faster hexagon counts updates when exploring new areas.

Finally, we have collected and processed several odometry datasets and we show the improvements in memory requirements and computational performance in single-story and multistory environments (Section 5.5).

The main contributions of this chapter have appeared in [60], [61] and [56].

5.1 Naïve FootSLAM Complexity Analysis

In the existing "naïve" implementation of FootSLAM (from now on called naïve FootSLAM), each particle's map:

$$\{M\}^{[m]} = \{\mathcal{H}_{h,l}, \{C_{h,l}\}^{[m]}\},$$

is implemented by means of a lookup table.

As explained in Section 4.2.2, FootSLAM's implementation of FastSLAM uses, in the limit, $N_{\mathcal{H}}$ separate estimators over the map. Thus, with N_p particles in the particle filter, FootSLAM maintains a total of $N_p \cdot N_{\mathcal{H}}$ map estimators.

The existing implementation of FootSLAM fixes the number of particles N_p. However, in the worst-case scenario, a person will visit a constant number of new prisms per time step. To cope with this situation, we envision a scalable implementation for FootSLAM in which the total number of prisms in the mapping area $N_{\mathcal{H}}$ can be dynamically incremented over time to allow the mapping of large areas. Thus, the number of prisms in the mapping area will be considered a linear function of time: $N_{\mathcal{H}} = \mathcal{O}(t)$.

Our goal is to analyze the computational complexity growth over time of the different steps of the RBPF algorithm (Section 2.5.8) given the input parameter $N_{\mathcal{H}}$: $T(N_{\mathcal{H}}) = T(t)$. Since N_p remains constant during the FootSLAM estimation process, the analysis with one particle is valid for any N_p. The complexity of the FootSLAM algorithm after time t, when the walk is over, can be obtained simply by summing the resulting complexity order per time step over all time steps.

Clearly, steps 2a, 2b, 2c and 2d of the RBPF (Section 2.5.8) require constant time at every time step k:

$$T_k^{\text{2a, 2b, 2c, 2d}}(t) = \mathcal{O}(1). \tag{5.1}$$

We will focus our attention on the last two steps of the RBPF algorithm (update and resampling steps) and analyze their dependency with the number of hexagonal prisms $N_{\mathcal{H}}$ and their impact on the complexity growth rate over time.

5.1.1 Complexity of the Update Step

The update step in naïve FootSLAM is related to the search and insertion costs of a lookup table, $\{M\}^{[m]} = \{\mathcal{H}_{h,l}, \{C_{h,l}\}^{[m]}\}$. This lookup table has been implemented using a hash table whereby collisions are prevented using a bijective correspondence between the hexagonal prisms and their hash code. Therefore, the time requirements of search and insertion are constant [33]:

$$T_k^{\text{upd}}(t) = \mathcal{O}(1). \tag{5.2}$$

5.1.2 Complexity of the Resampling Step

At the resampling step, each resampling particle needs to copy (with replacement) the state vector of the parent particle, including the map. Two cases can be easily differentiated when analyzing FootSLAM's complexity:

1. **Exploration of White Space**: By exploration of white space (ws) we refer to the stage in which particles explore new areas, visiting new hexagonal prisms. Thus, the size of the hash table grows linearly with time, making the resampling step require linear time in the number of visited prisms. As a consequence, the resampling step after an exploration phase depends on the time step k itself:

$$T_k^{\text{ws}}(t) = \mathcal{O}(\mathcal{N}_{\text{vis}}(k)) = \mathcal{O}(t). \tag{5.3}$$

2. **Revisiting Areas**: When the particles revisit prisms, the size of the lookup table remains constant, e.g. with $\mathcal{N}_{\text{vis}}$ entries. Therefore, the run-time of the resampling step will not depend on the time elapsed: Each particle needs to copy $\mathcal{N}_{\text{vis}}$ entries. In that case, the resampling step requires constant time:

$$T_k^{\text{rev}}(t) = \mathcal{O}(\mathcal{N}_{\text{vis}}) = \mathcal{O}(1). \tag{5.4}$$

As it can be observed, the complexity of naïve FootSLAM per time step is dominated by the resampling step during exploration phases, that is, FootSLAM requires time linear in the covered area for each time step k.

Hence, after integrating over all steps k we obtain the requirements in time per FootSLAM estimation process:

$$T^{\text{FS}}(t) = \mathcal{O}(t^2). \tag{5.5}$$

As a conclusion, we can state that the bottleneck of naïve FootSLAM is located at the resampling step when new areas are being discovered, showing a **quadratic-in-time** complexity growth rate.

FootSLAM's main goal is the mapping of large areas in quasi real-time applications [57]. The complexity analysis in this section indicates that a per-particle map data structure is not a suitable approach. In the next section we describe a new data structure that allows us to reduce the complexity growth rate of FootSLAM from $\mathcal{O}(t^2)$ to $\mathcal{O}(t \log t)$. This data structure is called *H-tree* and it is presented in the next section.

5.2 The H-tree: A New Data Structure for Complexity-reduced FootSLAM

The H-tree or Hexagon Tree is a tree-based data structure specifically designed for FootSLAM to reduce its time requirements from quadratic to linear times logarithmic, as suggested by [108]. Before further comments on the H-tree, a brief introduction to basic concepts of a tree-based data structure is summarized for clarity [33]:

1. A tree is an acyclic connected graph in which all nodes have one parent node except for the top node n_r (called root). Trees are usually drawn growing downwards.
2. All nodes have zero or more children nodes. Nodes that have no children are called leaves.
3. The degree of a node refers to the number of children of a node.
4. Any node can be reached following a unique path from the root node. The path is composed of edges or links.
5. The depth of a node is the length of the path that goes from the root to the node. The height of the tree refers to depth of the deepest node in the tree.

6. A balanced tree is a tree whose leaves are all located at the same depth.

The H-tree builds on the R-tree [68], that is, the H-tree is a geographic tree-based data structure. The R-tree is a balanced tree that uses rectangular shapes to quickly index geographical coordinates. Objects that are close to one another are grouped and represented by means of a bounding rectangle. Each node in the tree refers to a rectangle. A leaf node's rectangle refers uniquely to an object. The higher levels contain the aggregation of the lower levels.

The H-tree in FootSLAM is actually composed of two substructures: The Global H-tree (one per FootSLAM estimation process) and the Dynamic H-tree (one per particle). As we shall explain, the choice of such decomposition avoids the computational cost of constantly rebalancing the Dynamic H-trees during the mapping process - as opposed to the approach by [108], where only the dynamic structures are present.

In the following subsections, the Global and the Dynamic H-tree are presented in more detail.

5.2.1 The Global H-tree

The Global H-tree is a tree-based data structure that exists once and only once per FootSLAM estimation process. The Global H-tree is a tree-based data structure that stores all hexagonal prisms of the 3D mapping space, hence containing only geographic-related information: the coordinates of the hexagonal prisms, stored at the leaf level. The grid of prisms shown in Fig. 4.1 can be decomposed into two dimensions (left side of Fig. 5.1), generating two subtrees in the Global H-tree (right side of Fig. 5.1), a 2D-subtree and a 3D-subtree:

- *2D-subtree*: it stores and represents all N_H hexagons in the 2D grid (top left of Fig. 5.1) and has a height $H_{2D} = \log_s N_H$, where s is the degree of its internal nodes. Each "leaf" node corresponds to the root node of the 3D-subtree and has s_z children. The degree s of the nodes can be decomposed into two components: s_x and s_y, with the following geographical meaning: Nodes located at depth $D = (H_{2D} - 1)$ refer to groups of s_x columns of s_y hexagonal bases.
- *3D-subtree*: it stores and represents the N_z prism layers for any hexagonal base (bottom left of Fig. 5.1). Its nodes have s_z children, except for the leaf nodes, that have no children. Its height is $H_{3D} = \log_{s_z} N_z$.

The Global H-tree groups prisms that are located nearby. The root node n_r of the Global H-tree refers to all the hexagons (and prisms) in the grid. Lower levels refer to smaller groups of prisms. Both global and dynamic trees have in common the following characteristics:

- The maximum degree of all internal nodes is s in the 2D-subtree and s_z in the 3D-subtree.

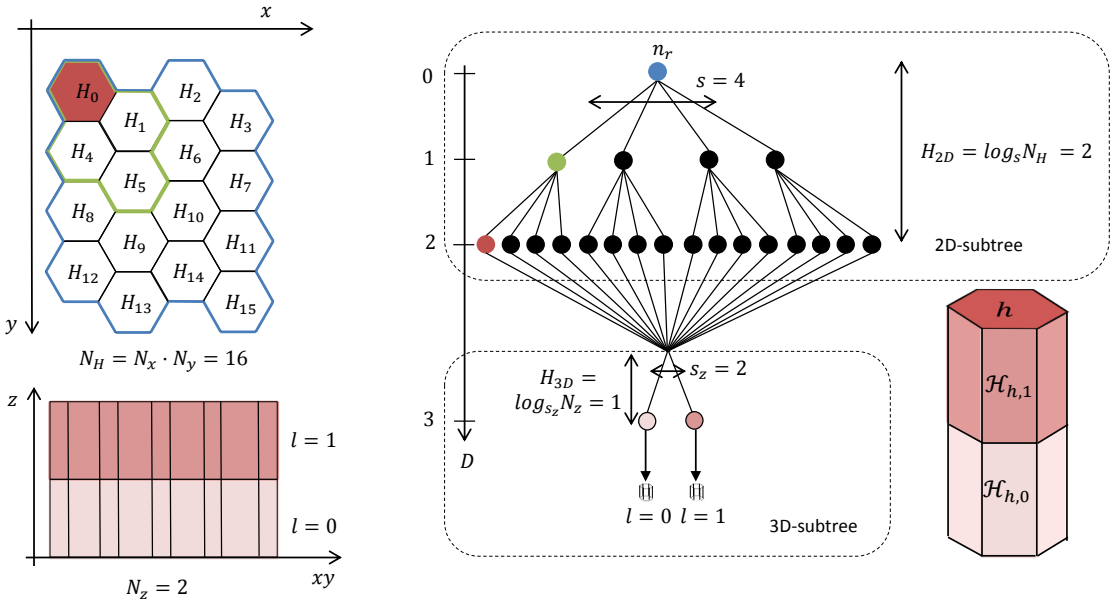


Figure 5.1: Construction of the Global H-tree. The hexagonal grid of Fig. 4.1 is composed of $N_x \cdot N_y = 4 \cdot 4 = 16$ hexagonal bases (top left) and $N_z = 2$ layers (bottom left). The resulting Global H-tree using a degree of $s = 4$ and $s_z = 2$ is shown on the right. The root node n_r of the Global H-tree (depth $D = 0$) refers to all $N_{\mathcal{H}} = 32$ prisms in the grid (marked in blue); each node one level below, up to depth $D = H_{2D} = 2$, refers to a group of prisms $s = 4$ times smaller (e.g. the green node refers to the prisms with base $\{H_0, H_1, H_4, H_5\}$, and the red node refers only to the prisms with base H_0). Finally the levels under $D = H_{2D}$ refer always to groups of prisms $s_z = 2$ times smaller, i.e. light brown refers to $l = 0$ and dark brown to $l = 1$. The prisms are stored at the leaf level.

- The tree is balanced: all leaves are located at the same depth.
- The height of the tree is $H_{\text{tree}} = H_{2D} + H_{3D}$.
- Generating and accessing a partial tree (e.g., the path to a leaf) can be done in time $\mathcal{O}(H_{\text{tree}}) = \mathcal{O}(\log_s N_H + \log_{s_z} N_z) = \mathcal{O}(\log N_{\mathcal{H}})$.
- Each leaf refers unambiguously to a hexagonal prism in the grid.

The path to a hexagonal base with index h, path(h), is composed of the links that lead from the root node n_r to the root of its 3D-subtree.

Remember that in the 2D grid the h index assigned to each hexagonal base was chosen to compute quicly the hexagonal base at which a particle is located given its 2D coordinates (Section 2.5.2): starting with 0 for the first row and first column and increasing the index for each column until there are no more columns in that row. These indices however do not allow for an efficient mapping of each hexagonal base to its path.

Thus, we store the paths to the hexagonal bases for quick indexing of hexagons and paths to each hexagon in the tree when a particle accesses its map:

$$\text{paths}_{\text{hex}} = \{H_h, \text{path}(h)\}, \quad 0 \leq h \leq N_H - 1. \tag{5.6}$$

Note that we do not need to store the paths from the hexagonal base to the corresponding layer because given a layer index $0 \leq l \leq N_z - 1$, the mapping to its corresponding path is defined by the layer index l. See more remarks at the end of this chapter.

Degree of the Nodes and Number of Nodes

The number of hexagons in the grid N_H is adjusted to be a power of s, or s to be a root of N_H. In order for the Global H-tree to be perfectly balanced and full, N_x must be a power of s_x and N_y a power of s_y for all nodes to have exactly s children. We choose $N_x = N_y$ and $s_x = s_y = s$ so that the only condition that must apply is that N_H be a power of s. Additionally, for the structure to become a tree, $s_x = s_y \geq 2$, i.e. $s \geq 4$. Furthermore, the degree of the nodes is upper bounded by the number of hexagons in the grid, so $4 \leq s \leq N_H$. Similarly, $2 \leq s_z \leq N_z$ and N_z must be a power of s_z to have full nodes in the 3D subtree.

In addition, the number of nodes N_n in the Global H-tree corresponding to the grid can be computed as follows:

$$N_n = \sum_{k=0}^{\log_s N_H} s^k + \sum_{j=0}^{\log_{s_z} N_z} s_z^j = \frac{s^{\log_s N_H + 1} - 1}{s - 1} + \frac{s_z^{\log_{s_z} N_z + 1} - 1}{s_z - 1} = \frac{s \cdot N_H - 1}{s - 1} + \frac{s_z \cdot N_z - 1}{s_z - 1}. \tag{5.7}$$

As a result, important considerations when choosing s are:

- The greater the s or the s_z, the fewer the nodes.
- The greater the s or the s_z, the greater the memory that needs to be allocated per node - given that each node is composed of an array of s or s_z pointers to the children nodes.
- The greater the s or the s_z, the shallower the tree. Since accessing and generating a partial tree has a cost in time $(\log_s N_H + \log_{s_z} N_z)$, the lower the access time is.
- As shall be shown, the greater the s or the s_z, the longer the run-time of the resampling step when using the H-tree.

5.2.2 The Dynamic H-tree

For each one of the N_p particles in the particle filter, we build a (sparse) Dynamic H-tree that contains the prisms that the particle visits along with their corresponding prism

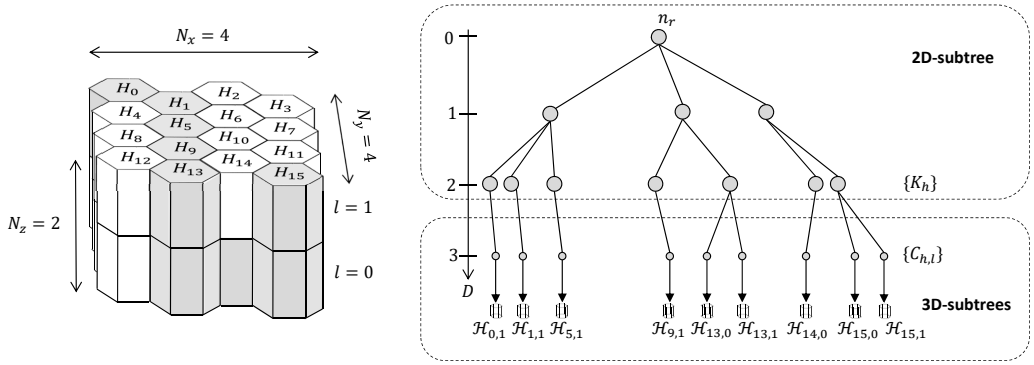


Figure 5.2: Dynamic H-tree of a particle that has visited nine prisms in two different layers (colored in gray on the left side). At the leaf level $D = H_{\text{tree}} = 3$ the prism transition counters $\{C_{h,l}\}$ are stored, whereas depth $D = H_{2D} = 2$ the aggregated counters for all layers, $\{K_h\}$, are stored.

transition counters $C_{h,l}$. Each Dynamic H-tree is based on the relative position between the nodes and links that lead to the prisms in the Global H-tree. Fig. 5.2 illustrates an example for a particle that has visited nine prisms. Note that in contrast to the Global H-tree, which is composed of one *single* 3D-subtree, the Dynamic H-tree contains a 3D-subtree *for each hexagonal base that the particle visits*. To achieve an efficient weight update step each Dynamic H-tree also stores, at depth $D = H_{2D}$ and for each visited hexagonal base, the aggregated transition counters K_h used in (4.8).

The first time the particle visits a prism, the path to its hexagonal base is obtained from the table of paths of the Global H-tree (5.6) and the corresponding nodes that lead to the hexagonal prism are copied into the Dynamic H-tree. If the particle revisits a prism, the Global H-tree is only needed to retrieve the path to the hexagonal base in the Dynamic H-tree.

In contrast to [108], using the Global H-tree as a reference, there is no need to constantly rebalance the Dynamic H-trees when new hexagons are stored. Nevertheless, this approach still allows for occasional redefinition of the trees, if we need to extend the mapping area as we explain in Section 5.3.

We have stated that each particle owns a dynamic tree. We shall see that some parts of a Dynamic H-tree (subtrees) can be shared by particles that have a common history of visited prisms. This is the strength of the H-tree and it is further explained in the next subsections within the context of the resampling and update steps of the FootSLAM algorithm.

5.2.3 Resampling Step in the Dynamic H-tree

As explained before, at the resampling step a set of N'_p particles is generated from the existing N_p particles based on their weights. A naïve approach duplicates the state information for all the particles that resample from a parent particle, as we showed with the hash table (Section 5.1.2).

The H-tree allows us to use another approach whereby all resampling particles share the information from the parent particle, avoiding duplicity of information: Equivalently to [108] the map of a resampling particle can initially be copied from a parent particle by merely copying the pointers of the root node of its Dynamic H-tree (maximum s). Figure 5.3 illustrates this concept for two resampling particles RP_1 and RP_2 that sample from a parent particle PP with a 2D grid.

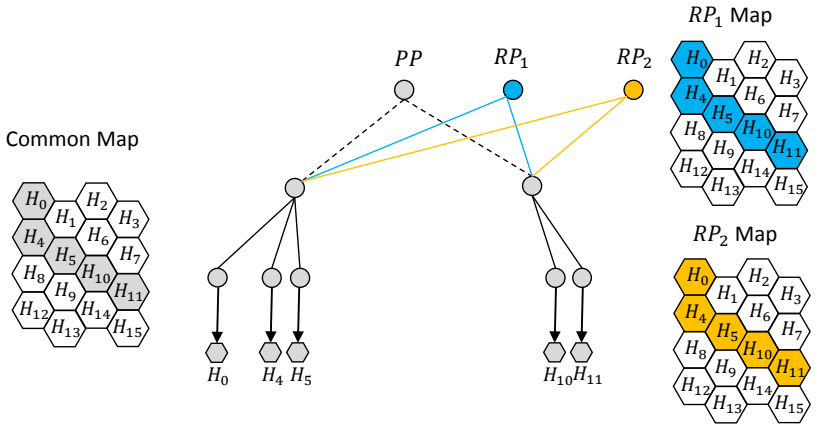


Figure 5.3: Resampling in the H-tree: Two resampling particles RP_1 and RP_2 resample from a parent particle PP by copying the pointers of its Dynamic H-tree root-node. RP_1 and RP_2 share the common hexagon map depicted on the left.

Operating in this manner, the resampling part of the algorithm does not depend on the number of visited prisms by each particle, but on the degree of the H-tree s, which is a constant value. The smaller the s, the shorter it takes to perform resampling.

5.2.4 Update of Hexagon Transition Counters in the Dynamic H-tree

When two or more particles resample from the same parent particle, they share the same nodes and the same visited prisms with their corresponding HTCs (hexagon transition counters). If one of these particles needs to update the HTC of a prism, then the path leading to it needs to be duplicated - with replacement [108]. Duplicating a path means duplicating its nodes and links. Figure 5.4 shows an example for particles RP_1 and RP_2 with a 2D grid.

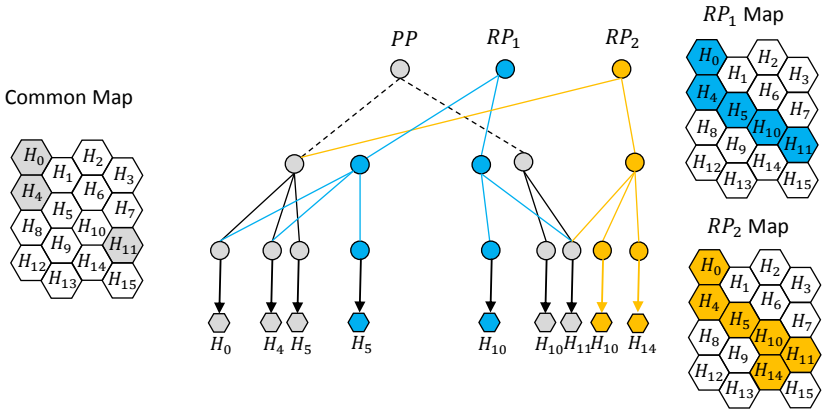


Figure 5.4: Update step in the Dynamic H-trees: After the resampling step, RP_1 exits H_{10} and revisits H_5. These hexagons' HTCs are common to particle RP_2 and cannot be directly updated. Instead, the paths that lead to them must be duplicated before the update to avoid changing the map of particle RP_2. Particle RP_2 visits a new hexagon (H_{14}) and needs to copy some of the nodes to avoid adding visited hexagons to RP_1. On the left, the common map to both particles after the update. On the right, the map that each particle possesses.

The complexity of the update step grows with the height of the tree H_{tree}:

$$\begin{aligned} T_k^{\text{upd}} &= \mathcal{O}(H_{\text{tree}}) &&= \mathcal{O}(\log_s N_H + \log_{s_z} N_z) \\ &= \mathcal{O}(\log N_H + \log N_z) &&= \mathcal{O}(\log(N_H \cdot N_z)) \\ & &&= \mathcal{O}(\log N_{\mathcal{H}}). \end{aligned} \tag{5.8}$$

In contrast to naïve FootSLAM, the update step using the H-tree shows a dependency on the number of prisms in the grid $N_{\mathcal{H}}$. In the worst-case scenario the mapping area grows over time, i.e. $N_{\mathcal{H}} = \mathcal{O}(t)$. As a result, $T_k^{\text{upd}} = \mathcal{O}(\log t)$. The total complexity of FootSLAM integrates over all time steps to:

$$T^{\text{FS}}(t) = \mathcal{O}(t \log t). \tag{5.9}$$

Note that when the number of prisms in the grid remains constant, the complexity of FootSLAM grows linearly with time, allowing real-time processing.

5.2.5 Memory Savings

A further advantage of using a tree-based data structure is the savings in memory, visible after the resampling steps. When resampling, N_p' new unique particles are generated from the old set of N_p particles by copying the pointers of the parent particle's root-node. If on average X particles are drawn from a given particle, then we experience a $1/X$ reduction in memory since we can reclaim the memory occupied by the obsolete particles. This effect

will be specially visible when loop closure takes place, that is, when the pedestrian revisits an area. In the limit, when all particles resample from one single surviving particle, we experience a ratio of $1/N_p$ savings in memory. Note that these memory savings are relative to the previous resampling step.

Figure 5.5 illustrates this idea over 4 resampling steps. At each resampling step some hypotheses are erased (marked with a cross) and memory is deallocated.

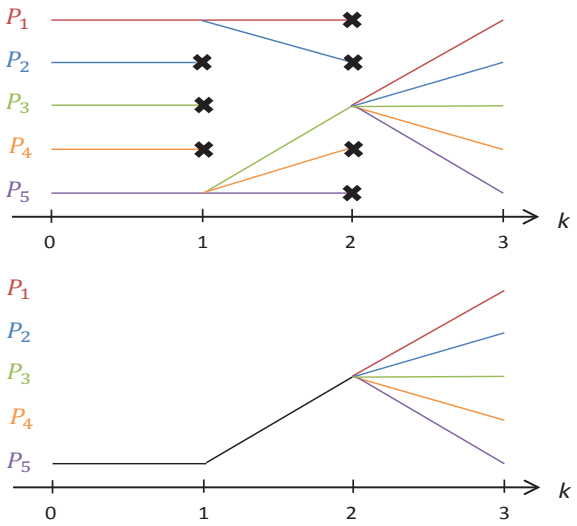


Figure 5.5: On the top, particles' hypotheses lifetime: Five different particles P_i are initialised at time step $k = 0$. At resampling time step $k = 1$, particle P_1 and P_2 resample from particle P_1. P_2 previous hypothesis is erased, marked with a cross. Likewise, particles P_3, P_4 and P_5 resample from P_5. At resampling time step $k = 2$, all particles are resampled from P_3. The only surviving hypothesis is that of P_3. On the bottom, the surviving history of hypotheses for the five particles is shown. At $k = 2$ the memory requirements have been reduced 5 times with respect to $k = 1$.

The reason for using a geographic approach for the H-tree is even clearer at this point: We live and work in buildings where large, unvisited spaces are rare. Therefore, our geographically defined Dynamic H-trees for a completed FootSLAM map will tend to become densely populated, except for perhaps the last (meter level) nodes. The underlying assumption which we believe that justifies this, is that for a given area, human accessibility is approximately uniform, except for meter level granularity.

If we assume that the space accessible to a pedestrian is not randomly distributed, but consists of groups of accessible areas (hexagonal prisms, in our case), then a geographic tree-based data structure will lead to a lower number of nodes needed in each Dynamic H-tree. Our Global H-tree is built grouping prisms that are nearby. Thus, visiting surrounding hexagonal prisms between two resampling steps translates into very local changes in the Dynamic H-trees. Only the nodes at the lower levels, i.e. a subpath, need to be generated. For example, suppose that particle RP_2 in Figure 5.4 visits H_{15} after H_{14}.

Then, adding hexagon H_{15} to the tree requires only adding one extra leaf node. As a consequence, in average less memory needs to be allocated than in the case in which prisms are randomly stored at the leaf level.

5.3 Size-adaptive FootSLAM based on the H-tree

FootSLAM uses a grid of hexagons as base for localization and mapping. This grid needs to be large enough to map the transitions of the pedestrian during the *whole* duration of her walk. With a fixed predefined hexagonal grid, a problem arises if the pedestrian walks out of the mapping area: her motion cannot be mapped in the form of face transitions, i.e. the map will be incomplete, and she cannot be located. Thus, we propose an approach to extend the original predefined hexagonal grid of N_H hexagonal bases according to the extent of the walk itself, allowing for size-adaptive FootSLAM. A similar approach may be applied to the 3D subtrees if new layers are visited.

5.3.1 Expanding the Hexagonal Grid

In the following let N_H^G denote the number of hexagons in the hexagonal grid that is predefined at the beginning of FootSLAM and N_H the total number of hexagons in the mapping area. Note that at the beginning of FootSLAM $N_H = N_H^G$. Additionally, let n_G refer to the node that corresponds to all the hexagons in the predefined grid and n_r refer to the root of the node, which are the same at the beginning.

As we shall explain, only the paths to the N_H^G hexagons in the predefined grid need to be stored.

A good criteria to expand this base grid is when any of the N_p particles visits one the hexagons one column or row away from the border. In the case of our example grid (Figure 5.1), when a particle visits any hexagon in column 2, we expand the grid to the left and when a particle visits any hexagon in column 3, the grid is expanded to the right. Similarly, if a particle visits any hexagon in row 2, then the grid is expanded on the top, and when a particle visits a hexagon in row 3, the grid is expanded on the bottom. Special cases are the corner areas, for example when a particle visits H_5 (row 2 and column 2), then the grid is expanded to the left, on the upper part and also on the top left corner.

A straightforward approach would generate a larger hexagonal grid and re-build the Global H-tree according to it. The problem with such an approach is that the memory requirements of the Global H-tree would grow with the total number of hexagons. In addition we would have the added computational requirement of re-building the whole Global H-tree and storing the path to each hexagon again. Thus, we propose an alternative that takes advantage of symmetries in any hexagonal grid.

Let us take our predefined hexagonal grid composed of $N_H^G = N_x \cdot N_y$ hexagons as base hexagonal grid. The mapping area is then expanded by attaching new hexagonal grids

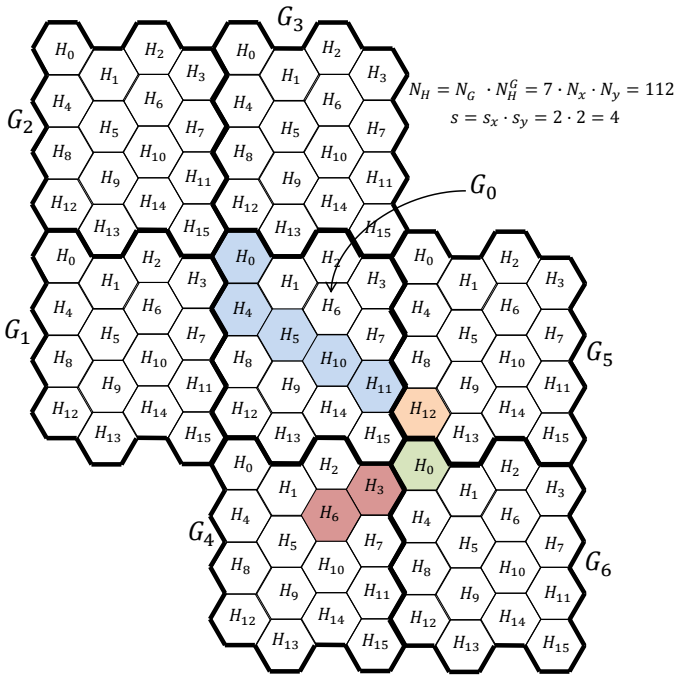


Figure 5.6: FootSLAM predefined G_0 as base hexagonal grid, with $N_H^G = 16$. The hexagons visited by a particle are marked with different colors. In this example, the particle moved from $H_0 \in G_0$ to $H_6 \in G_4$. When the particle visited $H_0 \in G_0$, the grids G_1, G_2 and G_3 were attached to G_0. Similarly, when the particle visited $H_{10} \in G_0$, then G_4, G_5 and G_6 were attached. Thus, the number of grids N_G is 7.

G_g of the same dimensions as the base grid where needed. An example has been depicted in Figure 5.6, where a particle moved from $H_0 \in G_0$ to $H_6 \in G_4$ triggering the extension of the matrix in different directions. Operating in this manner, the hexagons within each grid share the same layout and the total number of hexagons N_H can be computed as $N_H = N_H^G \cdot N_G$ where N_G is the number of grids.

As we shall see, the advantage of this approach is that we can control the growth of the grid of hexagonal prims and the Global H-tree, while making the complexity growth of FootSLAM only dependent on the logarithm of the number of grids N_G.

5.3.2 Expanding the Global H-tree

Expanding the Global H-tree to accommodate a new grid of hexagons can be done as follows: if the root n_r of the tree is full (i.e. the root has s children nodes), a new root node - also with degree s - is generated and the older root node is reattached to the new root node. If the root node is not full (i.e., there are fewer than s children nodes), a new path from the root node to n_G is created (Figure 5.7).

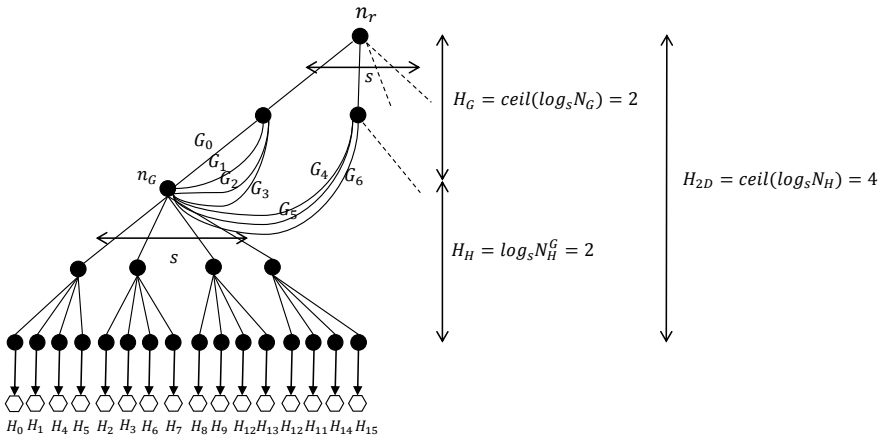


Figure 5.7: Extension of the 2D subtree of the Global H-tree of Figure 5.1 after attaching six new hexagonal grids $G_{1\cdots6}$ to the predefined hexagonal grid G_0 (Figure 5.6). The Global H-tree has grown two levels on the top.

This procedure is equivalent to having two subtrees within the 2D-subtree:

- A *grid subtree* with height $H_G = \lceil \log_s N_G \rceil$: the upper part of the tree, from the level of n_r to the level of n_G. This subtree is used to access the desired grid.
- A *hexagon subtree* with height $H_H = \log_s N_H^G$: the lower part of the tree (from n_G downwards). This subtree points to the hexagons within any grid.

As a result, the total height of the 2D-subtree is:

$$H_{2D} = H_H + H_G = \log_s N_H^G + \lceil \log_s N_G \rceil = \lceil \log_s (N_G \cdot N_H^G) \rceil = \lceil \log_s N_H \rceil. \tag{5.10}$$

The paths to each grid G_g, path(g), are now also stored for quick indexing of each grid in the tree when a particle accesses its map:

$$\text{paths}_{\text{grid}} = \{g, \text{path}(g)\}, \quad 0 \leq g \leq N_G - 1. \tag{5.11}$$

An advantage of this approach is that expanding the Global H-tree does not require us to rebuild the hexagon subtree or compute the paths to the hexagons again, but it only requires us to add occasionally some nodes in the grid subtree and add a new path to $\text{paths}_{\text{grid}}$ when a new grid is attached. Furthermore, the Global H-tree memory requirements over time will only grow with the number of nodes in the grid subtree, but not with the total number of hexagons. However, there will be additional memory requirements for the new data structure used to store the paths to the grids.

5.3.3 Rebalancing the Dynamic H-trees

Every time the Global H-tree adds one or more levels to its grid subtree, the N_p Dynamic H-trees need to be rebalanced by reattaching the root node of the Dynamic H-tree to the new root node.

From that moment on, the particles can visit hexagons in the other grids (for example the orange, green and red hexagons in Figure 5.6) and update their maps (Figure 5.8). To do that, each particle uses the Global H-tree to retrieve the path to the grid to which each hexagon belongs and the path to the hexagon within that grid (Equations (5.11) and (5.6) respectively).

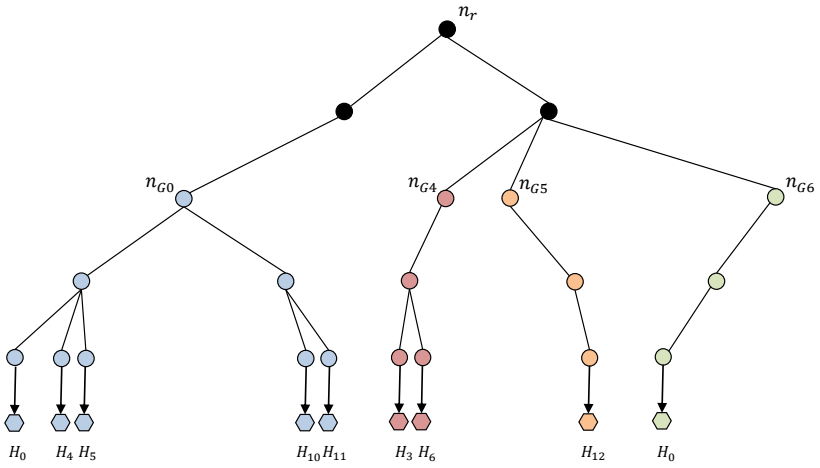


Figure 5.8: Dynamic H-tree of a particle that has visited $\{H_0, H_4, H_5, H_{10}, H_{11}\}^{G_0}$ (in blue), $H_{12}^{G_5}$ (in orange), $H_0^{G_6}$ (in green) and $\{H_3, H_6\}^{G_4}$ (in red). The paths to each grid and hexagon have been obtained from the Global H-tree in Figure 5.7.

5.3.4 Complexity Analysis of Size-Adaptive FootSLAM

The update step still requires time proportional to the height of the H-trees:

$$T_k^{upd}(t) = \mathcal{O}(H_{\text{tree}}) = \mathcal{O}(\lceil \log_s N_G \rceil + \log_s N_H^G + \log_{s_z} N_z), \tag{5.12}$$

which for a fixed number of hexagonal bases N_H^G and layers N_z per grid becomes:

$$T_k^{upd}(t) = \mathcal{O}(\log_s N_G) = \mathcal{O}(\log N_G) \tag{5.13}$$

In the worst case scenario, when ΔN_G new grids need to be added every time step to meet the mapping requirements, for example when no areas are being revisited and the pedestrian walks at a constant pace, then:

$$T_k^{upd}(t) = \mathcal{O}(\log t). \tag{5.14}$$

Hence, the complexity of FootSLAM remains the same with the adaptive-size algorithm.

As an example, suppose we have a 2D predefined grid of $N_x = 125$ times $N_y = 125$ hexagons with $s = s_x \cdot s_y = 5 \cdot 5 = 25$. If the hexagons have a radius $r = 0.5$ m, then

the area covered by one grid is $A_G \approx 100^2$ m^2. The height of the hexagon subtree is $H_H = \log_{25} 125 = 3$.

If our goal is to map an area A of say 3 km by 3 km, then we will need $N_G = A/A_G = 900$ grids by the end of the walk. The height of the grid subtree will need to grow from $H_G = 0$ to $H_G = \lceil \log_{25} 900 \rceil = 3$. As a consequence, even with a rather small base grid and a rather large area to be mapped, the computational cost of FootSLAM will grow slowly.

Note that if N_G remains constant through the whole FootSLAM estimation process, **FootSLAM requires linear time**, given that the $\log t$ factor is fixed by the constant height of the tree over time.

5.4 Compact Representation of Hexagon Transition Counters

Using concepts of source coding and lossless data compression [138], we can encode the different combinations of transition counts that the counters of an HTC (hexagon transition counters) can take on for any hexagonal prism $\mathcal{H}_i$. This codification is a compact representation (alphabet) for the set of transition counters across the faces of the hexagonal prisms.

The alphabet can map up to T transitions per face. Therefore, there are $(T+1)$ possible values (to account for 0 transition counts) for the eight counters to adopt and repetition is allowed. Hence the alphabet A is composed of $S = (T+1)^8$ symbols. The first symbol A_0 and last symbol A_{S-1} of the alphabet correspond, respectively, to:

$$\begin{aligned} A_0 &\leftrightarrow \{0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\} \\ A_{S-1} &\leftrightarrow \{T, T, T, T, T, T, T, T\}. \end{aligned}$$

Similarly to the Global H-tree, this alphabet is constructed at the beginning of the FootSLAM estimation process and exists only once. The alphabet is particularly effective during the exploration phases of FootSLAM that requires much particle diversity. As it shall be explained next, it will help reduce FootSLAM memory requirements.

5.4.1 Memory Savings

The alphabet is used to avoid storing the full HTC of the hexagons that the particles visit. Instead, the symbol corresponding to the HTC is stored at the leaf level of the particle's Dynamic H-tree.

The HTC data structure consists of a vector of length six. In our Java implementation it has been constructed using an array of eight Byte values, which requires 32 Bytes. On

$A_0 \leftrightarrow \{0,0,0,0,0,0,0,0\}$

$A_{0\to f=0}$	$A_{0\to f=1}$	$A_{0\to f=2}$	$A_{0\to f=3}$	$A_{0\to f=4}$	$A_{0\to f=5}$	$A_{0\to f=6}$	$A_{0\to f=7}$
$\updownarrow$	$\updownarrow$	$\updownarrow$	$\updownarrow$	$\updownarrow$	$\updownarrow$	$\updownarrow$	$\updownarrow$
$\{1,0,0,0,0,0,0,0\}$	$\{0,1,0,0,0,0,0,0\}$	$\{0,0,1,0,0,0,0,0\}$	$\{0,0,0,1,0,0,0,0\}$	$\{0,0,0,0,1,0,0,0\}$	$\{0,0,0,0,0,1,0,0\}$	$\{0,0,0,0,0,0,1,0\}$	$\{0,0,0,0,0,0,0,1\}$

Table 5.1: Example of symbol transitions given symbol $A_0 \leftrightarrow \{0,0,0,0,0,0,0,0\}$. Given a transition across any of the faces, the next possible symbols have been precomputed.

the other hand, each alphabet symbol is represented by a Short value, which requires only 2 Bytes.

Nevertheless, one should note that once a particle enters or leaves a hexagonal prism across a face whose counter had registered the maximum number of transitions, T, the symbol is converted back to an array of Bytes (HTC). The particle's Dynamic H-tree will then store the full HTC for that hexagon.

5.4.2 Efficient Map Updates

Say a particle is located at a hexagonal prism $\mathcal{H}_i$ whose associated set of transition counters $C_{h,l}$ corresponds to symbol A_i. The particle can leave the hexagonal prism using any of the eight faces. The particle can update the symbol by:

- Retrieving the corresponding $C_{h,l}$ of symbol A_i.
- Updating $C_{h,l}$ given the transition across face f: $C_{h,l\to f}$ as explained in Section 2.5.8.
- Obtaining the symbol $A_{i\to f}$ that corresponds to the updated HTC $C_{h,l\to f}$, if it exists.

However, for efficient map updates, for each alphabet symbol A_j the eight next possible symbols $A_{j\to f}\ \ \forall f$ are precomputed and stored during the construction of the alphabet. Table 5.1 represents an example for symbol A_0.

Operating in this manner, the update of a symbol requires only to know which face was crossed when leaving the hexagonal prism, avoiding explicit conversion to an HTC.

During exploration phases, when particles are visiting new hexagons and there exists much diversity of hypotheses, memory requirements grow with the visited area. The alphabet will play a significant role during those phases, reducing the rate of growth of the memory requirements.

5.5 Experimental Verification

5.5.1 Methodology

Walk Covering a Large 2D Area

FootSLAM has been specifically developed to map indoor environments. However, to evaluate the reduction in complexity of the proposed map structure a pedestrian undertook a walk outdoors - but without satellite navigation - covering a relatively large 2D area (ca. 100 m by 100 m) area. The walk lasted over 13 minutes and the step measurement data were collected using an MTx - Xsens® IMU sensor attached to her shoe. The data collected (Figure 5.9(a)) were processed single threaded on an Intel® Xeon® processor with a clock speed of 3.46 GHz.

In the FootSLAM algorithm we use $N_p = 30\ 000$ particles and a single hexagonal grid composed of $N_H^G = N_x \cdot N_y = 256 \cdot 256$ hexagonal bases with $N_z = 1$. We will study the performance of FootSLAM by varying the maximum number of mapped transitions by the alphabet, T, and the degree of the nodes in the Global H-tree.

Figures 5.9(b) and 5.9(c) show the FootSLAM maps corresponding to the odometry data. Clearly, FootSLAM was able to correct the drifting errors visible in the odometry data. A video showing the evolution of FootSLAM's estimation process can be found in [25].

Walk in a Multistory Environment

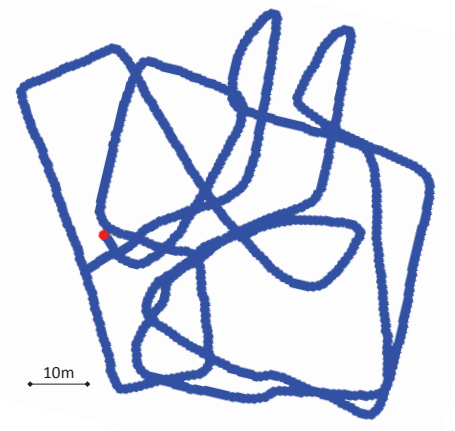
For a further validation of the proposed algorithm, we also processed the multistory dataset collected during Walk 3 of Section 4.6.2 with $N_x = 256$, $N_y = 256$, $N_z = 64$, $s = 16$, $s_z = 2$ and $N_p = 30\ 000$. We will show the evolution of the number of nodes during the walk as an indicator of the memory improvements.

FootSLAM has been implemented on a Java platform. Next, the constraints that the implementation presents for T and s are explained.

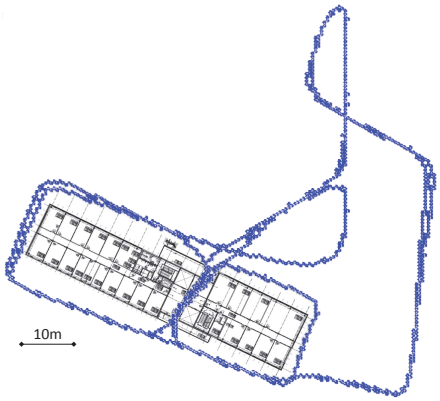
5.5.2 Maximum Number of Mapped Transitions by the Alphabet

We represent alphabet symbols by means of a Short type, which in Java covers a range of $(2^{16} - 1)$ possible values. Consequently the number of symbols in the alphabet S is bounded by this value and the maximum value that T can take on for 8 faces is: $T = \lfloor \log_8(2^{16} - 1) \rfloor - 1 = 4$. However, when mapping a 2D environment, we only allow transitions across the 6 lateral faces. Thus the maximum value that T can have is: $T = \lfloor \log_6(2^{16} - 1) \rfloor - 1 = 5$.

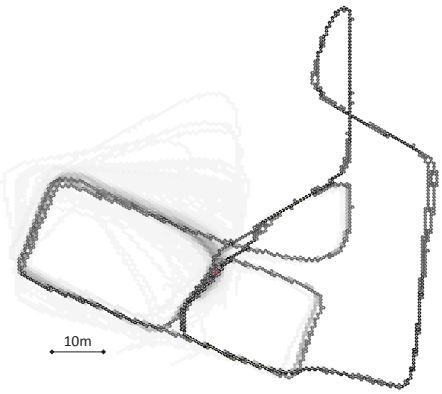
From the moment when a particle has crossed one of the faces of a hexagon more than that limit, the symbol is converted to an instance of HTC.



(a) Odometry.



(b) Maximum a posteriori map.



(c) Cumulative map.

Figure 5.9: Odometry recorded during the walk (Figure 5.9(a)) and FootSLAM output maps: map corresponding to the maximum a posteriori hypothesis (hypothesis of the particle with the highest likelihood) for the history of pedestrian's poses (Figure 5.9(b)) and cumulative map with the aggregated information of all particles (Figure 5.9(c)). Note that the building layout is only shown as a reference. The building geographic coordinates are: (48.084758, 11.277903).

Of course one can think of a more sophisticated alphabet that allows larger counts for the lateral faces than for the upper and lower faces (which have a lower probability of being crossed).

5.5.3 Limit for the Degree of the Nodes

We use a Byte representation for the links of each node, so we must limit the degree s of the nodes to be 255 or fewer. Applying all conditions stated in Section 5.2.1, the only two values we can use for the H-tree are: $s = 4$ and $s = 16$, which correspond to a tree height of $H_{\text{tree}} = H_{2D} = 8$ and $H_{\text{tree}} = H_{2D} = 4$ respectively.

Note that we have chosen to encode the path using a vector of Byte values due to the considerable number of paths to be stored (N_H^G) when the Global H-tree is generated. Using a Short type or an Integer type is also possible if the available memory allows it. Alternatives to lower the memory requirements of the H-tree are left for future work (see Section 5.6.1).

5.5.4 Time Performance

In the following we will use the term naïve FootSLAM to refer to the implementation of FootSLAM of Section 5.1 and H-tree FootSLAM for the complexity-reduced implementation based on the H-tree. Naturally, the resulting map is the same no matter which approach is used.

Figure 5.10 shows the measured time performance of naïve FootSLAM (red curve) and H-tree FootSLAM for two different degree values: $s = 4$ (pink line) and $s = 16$ (blue line for $T = 5$ and green line for $T = 0$, which are overlapping).

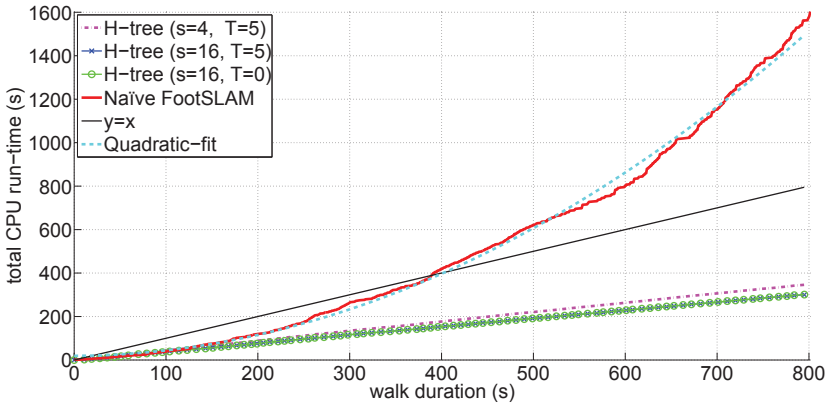


Figure 5.10: Measured run-time of the FootSLAM algorithm for the walk depicted in Figure 5.9 for different FootSLAM implementations: naive FootSLAM (red curve) and H-tree FootSLAM (dash-dotted pink line, blue line with cross markers and green line with circle markers). Real-time processing is depicted by a black line. The dashed cyan curve represents the best fitting quadratic curve to the performance of naive FootSLAM.

Our experimental results are consistent with the theoretical analysis of FootSLAM: Naïve FootSLAM shows a quadratic in time complexity growth rate (the dashed cyan curve corresponds to the best quadratic-fitting curve). On the other hand and as discussed in Section 5.3.4, H-tree FootSLAM requires time in $t \log t$, which for a single grid composed of a fixed number of hexagons N_H^G, results in linear time complexity growth rate.

The use of a tree-based data structure like the H-tree shows a clear advantage over the previous naïve FootSLAM implementation. After ca. 400 seconds, naïve FootSLAM is

not able to perform in real-time. On the other hand, the time requirements of H-tree FootSLAM stay always under the real-time processing limit (black line).

The use of different values for the degree of the nodes gives the same order of complexity growth rate (linear times log), but changes the speed of growth (the slope). In this case, a greater degree value (blue and green lines, corresponding to shallower trees) improves the performance of H-tree FootSLAM. Such result does not imply that the performance will always improve for a greater degree value: A greater s will also slow down the resampling step.

As could have been expected, using the alphabet does not improve the performance in terms of run-time: for $T = 0$ (no alphabet) and $T = 5$ the performances are practically the same (the blue and green line overlap). Our approach of storing the possible symbol transitions (Section 5.4.2) has allowed the alphabet-based implementation to maintain the speed of the update step of the implementation with no alphabet. The advantage of using the alphabet is visible in terms of memory requirements, as we shall show next.

5.5.5 Memory Performance

Figure 5.11 depicts the measured memory performance of naïve FootSLAM (red curve) and H-tree FootSLAM for two degree values, $s = 4$ (pink and black curves) and $s = 16$ (blue and green curves). In contrast to Figure 10 in [60], where the effect of Java's garbage collector was visible, Figure 5.11 shows the minimum memory requirements for each implementation (as a result of connecting the local minima after garbage collection).

As argued at the end of Section 5.2.5, H-tree FootSLAM performs better than naïve FootSLAM in terms of memory requirements. This is visible in Figure 5.11 since the memory requirements of naïve FootSLAM increase almost monotonically over time, whereas the memory needed for the H-tree implementation remains bounded. The highest memory requirements of H-tree FootSLAM correspond to ca. 1050MB (green curve, $s = 16, T = 0$), while naïve FootSLAM highest memory requirements correspond to ca. 2525MB.

On the other hand, given a degree value s, the use of the alphabet (blue curve for $s = 16$ and pink curve for $s = 4$) improves the memory performance of the H-tree implementation without an alphabet (green curve for $s = 16$ and black curve for $s = 4$). The improvements are specially important during exploration phases, i.e. between $t = 0$ and $t = 220$ seconds, between $t = 375$ and $t = 450$ seconds and between $t = 475$ and $t = 625$ seconds.

Furthermore, given a fixed maximum number of precomputed transitions ($T = 5$), the memory performance of H-tree FootSLAM for $s = 4$ (pink curve) improves over that of H-tree FootSLAM with $s = 16$ (blue curve). This is because of the fact that even though the number of nodes that need to be generated to insert all visited hexagons is lower for $s = 16$ than for $s = 4$ (Equation (5.7)), each node of degree $s = 16$ requires more memory. This result does not imply that lower degree values will always perform better in terms of memory, since there is always a trade off between the number of nodes allocated in memory and the size of each node.

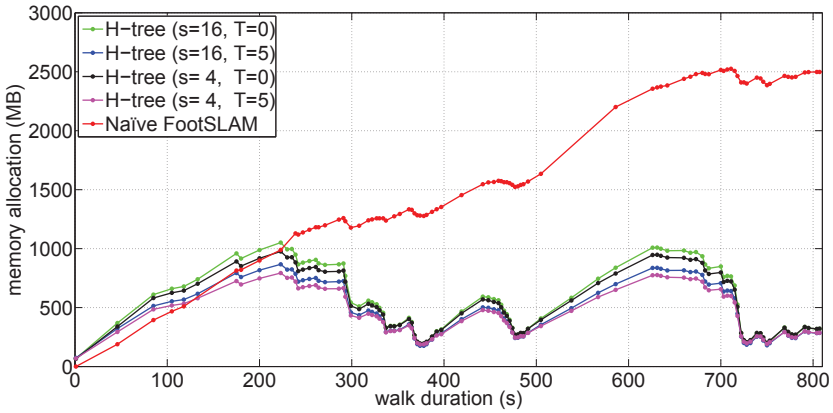


Figure 5.11: Measured memory performance of naive FootSLAM (red curve) and H-tree FootSLAM with degree values $s = 4$ (dash-dotted pink curve) and $s = 16$ (green curve with circle markers ($T = 0$) and blue curve with cross markers ($T = 5$)) for the walk depicted in Figure 5.9.

Number of Nodes

We can also measure the efficiency of our H-tree implementation in terms of the number of tree nodes allocated in memory during the walk. Figure 5.12 shows the total number of nodes for all particles, that is, the sum of the number of nodes used in the $N_p = 30000$ Dynamic H-trees, compared to the number of actually allocated nodes in memory. As discussed in Section 5.2.5, when the pedestrian closes a loop, a greater number of particles is resampled from a few particles and the memory requirements decrease, visible in a reduction in the number of allocated nodes in memory. As expected, a shallower tree ($s = 16$) requires to allocate fewer nodes in memory than in the case of a deeper tree ($s = 4$).

Fig. 5.13(b) illustrates the impact of the H-tree in multistory environments. The dataset was collected during Walk 3 of Section 4.6.2. In Fig. 5.13(b) we have depicted the total number of nodes for all particles divided by 3 (in red), and the total number of nodes actually allocated in memory during the FootSLAM estimation process (in blue).

After about 100 s, the pedestrian visited a corridor that he had previously visited but one floor below, acting as a prior (point A in Fig. 5.13(a)). At roughly 300 s the pedestrian went back along a corridor (point B), closing a loop. Despite loop closure, the number of total nodes grows monotonically except between 600 s and 1400 s, while the pedestrian was standing (area C). Afterward the pedestrian kept walking and exploring. Point D marks a path out of the building and point E corresponds to the exploration of a ramp; both these loops represent novel areas associated with a strong increase in the total number of nodes whereas the memory requirement falls after each loop closure. The evolution of the blue curve is nearly constant because due to resampling particles are sharing a large proportion of the map.

The H-tree has also been successfully applied to efficiently map the magnetic field in

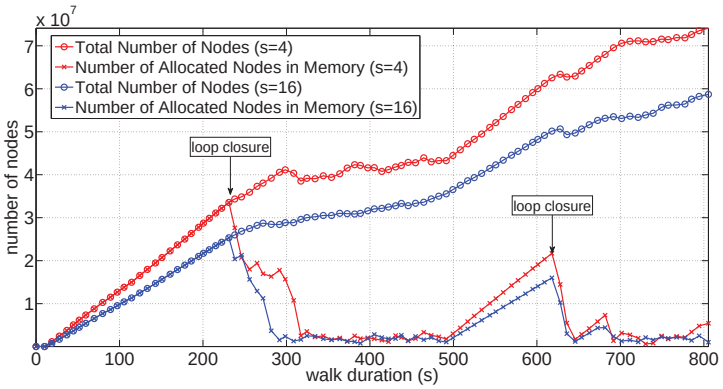


Figure 5.12: Total number of tree nodes for all particles (circle marker) and number of allocated nodes in memory (cross marker) needed during the walk depicted in Figure 5.9 for $s = 4$ (red curves) and $s = 16$ (blue curves). At roughly $t = 230$ and $t = 610$ seconds the pedestrian started to close two big loops. After the subsequent resampling steps, a greater number of particles share common parts of their Dynamic H-trees, visible in the reduction in the number of allocated nodes in memory.

multistory environments [134]. Two videos that show the particle filter estimation process in two different buildings are available online [132][135].

5.6 Discussion

In this chapter we have studied the complexity of a naïve FootSLAM implementation. The resampling step represents the bottleneck, showing a linear dependency with the covered area per time step. Such dependency makes FootSLAM exhibit a quadratic-in-time complexity growth rate that prevents real-time processing.

To address this, we have presented a new geographic tree-based data structure, called H-tree, reducing FootSLAM's computational growth rate in time from quadratic to linear times logarithmic. This data structure also reduces the memory footprint of FootSLAM, specially after resampling, since particles that share a common history of visited areas can now share the same local maps.

In addition, an alphabet of symbols has been introduced to compactly represent map estimates (hexagon transition probabilities) by the particles. The alphabet reduces the memory requirements of FootSLAM especially during exploration phases in which much particle diversity of hypotheses is needed.

The following subsections discuss additional improvements to the techniques presented in this chapter.

5.6.1 H-tree Memory Improvements

Storing the paths to each hexagonal base (5.6) could be avoided by assigning a second index to each hexagonal base (see Fig. 5.14).

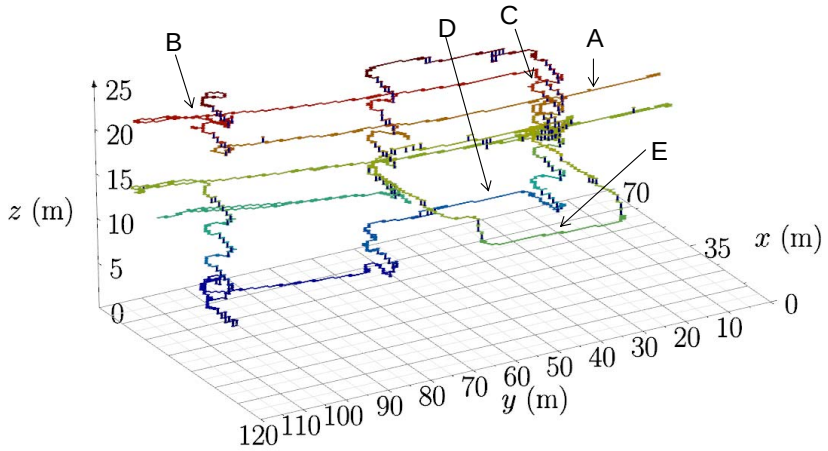
The memory requirements of assigning a new index to each one of the N_H hexagonal bases is, if using an Integer (4 Bytes): $4N_H$ Bytes. In contrast, the table of paths needs N_H entries each with a path of $\log_s N_H$ Bytes. The impact of these memory savings will be greater the larger the number of hexagons in the grid N_H is.

Note: we still need to store the grid indices used throughout this thesis for quick indexing of the hexagons, which is critical to determine what faces were crossed by the particles.

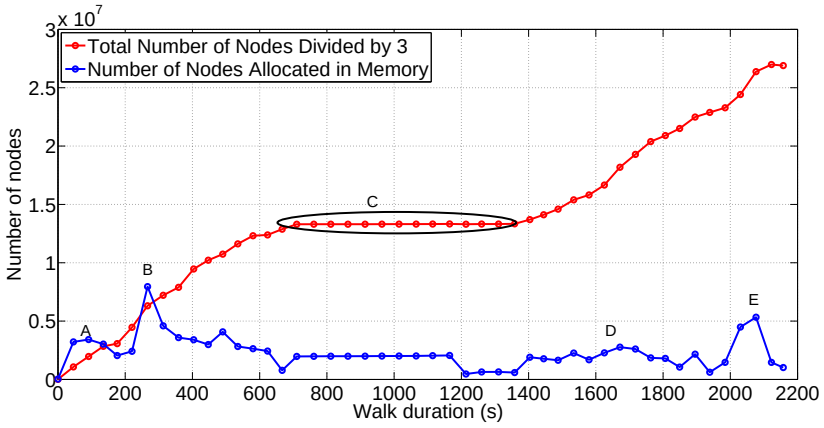
Further work should explore ways to reduce the memory footprint of the H-tree and the grid of hexagonal prisms.

5.6.2 Closing Larger Loops

The algorithms and data structures presented in this chapter allow us to use FootSLAM in real-time while mapping *larger* areas. However, due to computation and memory limitations, the mapping area (the hexagonal grid) cannot grow indefinitely. We could apply a similar approach to Hierarchical SLAM [46], whereby the map is sliced into local maps as the subject (in this case a robot) moves, and then a higher global map level system imposes loop closure constrains to join and fuse the local maps.



(a) Resulting FootSLAM map from Walk 3 in Section 4.6. We have labeled with letters interesting locations for which the number of H-tree nodes (Fig. 5.13(b)) shows interesting behavior.



(b) Total number of Dynamic H-tree nodes for all 30 000 particles (red) and number of allocated nodes in memory (in blue) for Walk 3 (Fig. 5.13(a)) using s=16 and $s_z = 2$. We have divided the total number of nodes by 3 to be able to distinguish interesting events (labeled with letters) in the blue curve. Thanks to the H-tree, the particles share the storage resources for common parts of the map. See main text for a comparative description.

Figure 5.13: Example of the reduction in memory resources needed when using the H-tree in a multistory environment.

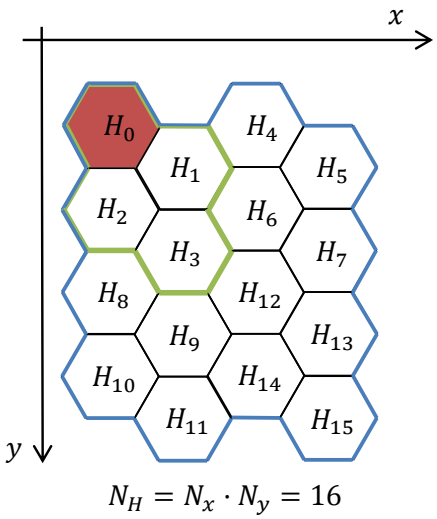


Figure 5.14: Alternative indices assigned to the hexagonal bases.

6 Reducing the Complexity of FeetSLAM's Map Combination Step

The FeetSLAM algorithm, introduced in Section 2.6, combines the resulting maps from several walks to yield more accurate and complete maps that can be used for reliable indoor pedestrian navigation. In Chapter 3 we derived an entropy-based metric to measure the uncertainty of maps that can be used to select the order in which a set of maps is combined, reducing the complexity of FeetSLAM from quadratic to linear in the number of datasets. In Chapter 4 we extended FootSLAM to map multistory environments and in Chapter 5 we have presented an approach to reduce its computational complexity and memory requirements towards real-time estimation.

The remaining challenge towards real-time collaborative mapping with FeetSLAM is to reduce the complexity of the map combination step. The goal is to quickly find the geometric transformation between any pair of maps so that they can be combined to one map. This chapter proposes an alternative algorithm to the transformation search of [136], which performs a brute-force search over rotation, translation and scale.

Our proposed algorithm computes the angular spectrum (using the Hough transform) and angular spectra of two FootSLAM maps and correlates these spectra to find candidates for the rotation and translation, respectively, that places the maps within the same coordinate system. This approach reduces the complexity of the map combination step from quadratic to linear in the explored area of a floor.

We will show the results of combining 2-6 maps resulting from data collected in three single floor environments and one multistory environment. The average time to combine two maps has been reduced from tens of minutes to 4 seconds.

The main contributions of this chapter have been submitted to a journal [59].

6.1 Problem Statement: Fast Map Combination Step

To formulate the problem, let's assume that we can represent the FootSLAM probabilistic motion map of a building floor by a (finite) set of n points:

$$\mathcal{P} = \{\{p_0, \bar{C}_{p_0}\} \ldots, \{p_i, \bar{C}_{p_i}\}, \ldots, \{p_{n-1}, \bar{C}_{p_{n-1}}\}\}, \tag{6.1}$$

where $p_i \in \mathbb{R}^2$ represents a point in 2D Cartesian space and $\bar{C}_{p_i} \in \mathbb{R}_{[0,1]}$ encodes the local probability of a certain angular transition at that coordinate as learned by the FootSLAM algorithm [128].

When using only inertial sensors, i.e. when no global coordinate system is available, FootSLAM probabilistic maps are rotation and translation invariant. As a consequence, given two FootSLAM maps represented by point sets $\mathcal{P}$ and $\mathcal{Q}$, respectively, we are interested in finding the geometric transformation T that we need to apply to $\mathcal{Q}$ to best match the point set $\mathcal{P}$ so that we can add their contributions to compose a combined map.

Finding the geometric transformation between two point sets is a classical problem in computer vision and is known as *point set matching*. Our problem falls into the category of point set matching with unknown correspondence, since we only assume that the individual maps (and thus the resulting point sets) overlap to certain degree, for example in a scenario where multiple pedestrians roam throughout the same building but enter from different locations.

In our case T is a scale-preserving isometry, also known as rigid transformation, which preserves the Euclidean distances between every pair of points. For simplicity, note that in this contribution we are not optimizing for scale, because our datasets collected with a foot-mounted sensor array [136] seem to share a similar scale factor.

Given a point $q_i = (x, y) \in \mathcal{Q}$, we define $T(q_i, \Delta\theta, \Delta x, \Delta y) = (x^T, y^T)$ with:

$$\begin{pmatrix} x^T \\ y^T \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\Delta\theta & -\sin\Delta\theta & \Delta x \\ \sin\Delta\theta & \cos\Delta\theta & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{6.2}$$

which corresponds to a planar counterclockwise rotation of $\Delta\theta$ about the origin of the coordinate system followed by a translation $(\Delta x, \Delta y)$. When merging maps from multi-story buildings, we assume that the displacement along the vertical axis Δz is known (or can be inferred with an independent method, for example using an altimeter).

Remember that the underlying structure of FootSLAM maps is a grid of hexagonal prisms (see Chapter 4). To take care of the possible mismatch between the transformed point set $\mathcal{Q}^{\mathcal{T}}$ and $\mathcal{P}$, before adding the two point sets, M^{Q^T} is projected onto the same grid as M^P (more details in Section 6.4.1).

6.1.1 Contributions

The approach proposed in this chapter adapts and extends Carpin's work in [28] to FootSLAM maps. Carpin proposes merging two occupancy grid maps – obtained by two robots that carry laser scanners – by converting them first to binary images. Then, his proposed algorithm computes for each map its angular spectrum (using the Hough

Transform (HT)) and spatial spectra along the x and y axes. This simplifies the transformation search problem by dividing it into three separate problems: finding $\Delta\theta$, Δx and Δy, respectively.

The main novelty of our work with respect to Carpin's lies in the nature of the maps to merge: occupancy grid maps obtained with laser scans intrinsically reflect the presence of linear features such as walls, which is a pre-requisite to compute the angular spectrum of a map based on the HT; FootSLAM probabilistic maps encode the local probabilities of human motion at the visited areas. With the particular case of FootSLAM maps, we show how one can still apply the HT to any human motion based map, given that human motion also exhibits linear features: pedestrians tend to walk following straight lines and avoiding obstacles, for example when walking in a corridor or crossing a door. Our approach does not require to convert the maps to pixels: we work directly with a suitable representation of the probabilistic map that lends itself to the HT and the spatial spectra in order to identify the transformation between two maps. Furthermore, FootSLAM maps are the result of processing data collected by walking pedestrians using only low-cost inertial sensors instead of laser range-finders. This poses two major advantages:

a) since many pedestrians in the modern world already walk on a daily basis carrying smartphones with inertial sensors, the effort to scale the mapping task to the whole indoor world may be conducted in a passive manner and without having to deploy robots, and

a) inertial sensors pose fewer privacy concerns than laser range-finders because they reveal no critical information about the environment or other humans.

Finally, in this chapter we also study the complexity of the algorithm and apply the algorithm to three single-story environments and one multistory environment.

The rest of this chapter is organized as follows: Section 6.1.2 reviews related work in the field of point set matching. Section 6.2 introduces the HT and defines the Hough Spectrum (HS) and Section 6.3 explains how to obtain the angular and spatial spectra of a FootSLAM map. Section 6.4 explains how to combine two FootSLAM maps using their angular and spatial spectra. To this end, we show how to extend the projection of FootSLAM maps to multistory environments. Next, Section 6.5 studies the computational complexity growth of the proposed algorithm and Section 6.6 describes the experiments and the results.

6.1.2 Related Work on Point Set Matching

Many different algorithms tackle the problem of aligning 3D point clouds [15].

The Iterative Closest Point (ICP) algorithm [18] iterates to find the geometric transformation that minimizes the difference between the points of a source point cloud and those of a target point cloud. To achieve convergence, the ICP algorithm requires the two

point clouds to be *similar*, for example, if these point clouds come from two close-in-time successive scans by a robot carrying a laser scanner with low noise.

The Largest Common Point Set (LCP) family of algorithms, popular in the field of Biology, attempts to identify substructures within two 3D point clouds (for example those of two proteins) [31][4]. In the worst-case scenario, the complexity of the algorithm runs in $\mathcal{O}(n^{7.5})$, being n the number of samples in the point set to transform.

In the field of object recognition Belongie et al. [16] present a novel approach to deal with non-rigid transformations and unknown correspondences: for each point in one point cloud, the authors propose finding its corresponding point in a second point cloud by looking at their *shape contexts*. The complexity of the proposed algorithm exhibits quadratic and cubic growth in the number of points in each shape. This approach remains a promising alternative not investigated in this thesis.

Hough introduced the Hough Transform (HT) in a patent filed in 1962 [73] as a technique to recognize complex patterns. The HT has been widely used in computer vision and image processing to extract lines and curves [44] and other shapes (see [76] for a survey on the topic). In the robotic community, the Hough Scan Matching [30] is a non-iterative, deterministic 2D scan matching method based on the HT that is able to track multiple solutions in unstructured environments using the angular spectrum of two successive scans. In [28], the idea of the angular spectrum based on the HT is extended by defining the spatial spectrum to quickly find the geometric transformation that aligns two 2D occupancy grid maps generated by two robots that explore the same environment carrying laser scanners. Once aligned, the two maps can be combined to generate a more extensive map.

6.2 The Hough Transform and the Hough Spectrum

The maps that are the result of FootSLAM will tend to well represent the major structures of the indoor environment, like corridors. Hence, we are motivated to focus on detecting lines given that walls usually constrain pedestrians to walk following straight segments in indoor environments. These detected lines can be used in turn to infer the main angular contributions of a walk. Although there exist other algorithms that may be more time efficient to detect lines, the HT is a robust algorithm to deal with noisy point clouds.

6.2.1 The Hough Transform (HT)

The HT parametrizes a straight line in the Cartesian plane $\mathbb{R}^2$ (*input space*) using [44]:

$$\rho = x\cos\theta + y\sin\theta, \tag{6.3}$$

where (x, y) represent the Cartesian coordinates of the points along the line, and ρ and θ are the parameters of the *polar parameter space* (Hough space) also in $\mathbb{R}^2$. The parameter

ρ corresponds to the normal distance from the center of the Cartesian coordinate system to the line; the parameter $\theta\,(0 \leq \theta \leq \pi)$ denotes the angle between the x axis and the normal line (Fig. 6.1). Thus, *a straight line in the input space corresponds to a point in the parameter space.*

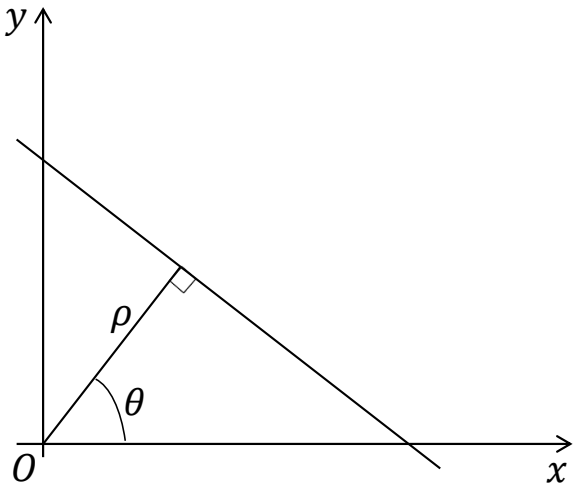


Figure 6.1: Polar parametrization of a straight line.

To detect lines, we proceed as follows [44]: For each point (x, y) in the input space (a matrix of dimensions $h \times w$) we compute the corresponding pairs (ρ, θ) using (6.3). To lower the computational burden of the algorithm, both θ and ρ are quantized using, respectively, N_θ bins of size $\Delta_\theta^q = \pi/N_\theta$ rad and N_ρ bins of size Δ_ρ^q m with $N_\rho = \max\{h, w\}/\Delta_\rho^q$ and period π. A bidimensional accumulator array accumulates the number of appearances for each bin. The local maxima in the accumulator indicate the presence of the main angular contributions (lines) in the input space. As we shall show, applied to human motion within a building, it can reflect the presence of long linear structures, such as corridors.

6.2.2 The Hough Spectrum (HS)

Given the HT accumulator array of an input space, Censi et al. define the HS as follows [30]:

$$\mathrm{HS}(\theta_j) = \sum_{i=0}^{N_\rho - 1} \mathrm{HT}(\theta_j, \rho_i)^2, \quad 0 \leq j \leq N_\theta - 1. \tag{6.4}$$

with a period of π rad. The HS represents the angular spectrum of the input space, i.e. the distribution of angles in the input space, which is independent of translations along the x and y axes. Shortly, we will show how this property may be used to identify the main linear contributions of human motion within a building.

In the following we will introduce the Hough Transform in the context of FootSLAM maps, the Hough Spectrum as a representation of the angular features of a map, and

spatial spectra as a representation of the location of the main features of a map along the axes of a 2D Cartesian coordinate system.

6.3 Applying HT and HS to FootSLAM Probabilistic Maps

6.3.1 Angular Spectrum of FootSLAM Maps based on the Hough Transform

The angular spectrum of a FootSLAM map shall reflect the orientation of the main linear motion contributions. In this section we show how to apply (6.4) to a single layer of hexagonal prisms.

In the image processing domain the input space of the HT comprises an image's pixels [44]. That is why Carpin [28] transforms the rich laser-scanner based occupancy grid map before any further processing occurs. However, probabilistic motion maps are fundamentally different from occupancy grid maps, with each cell encoding 8 face transition probabilities. We need to find suitable representation for the hexagonal map that lends itself to the HT. We choose our input space $\mathcal{P}$ (6.1) to encompass the center points of the lateral faces of the visited hexagons (Fig. 6.3), with each center point encoding the count ratio of that face: $\bar{C}^f_{h,l}$, defined as:

$$\bar{C}^f_{h,l} = \frac{C^f_{h,l}}{C_{h,l}}, \tag{6.5}$$

which can be interpreted as a local transition probability.

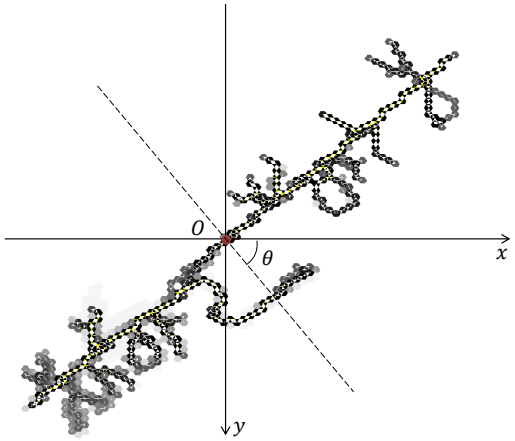
To apply the HT, we set an arbitrary point within the map as center of rotation (marked in red in Fig. 6.2(a)). For each layer L_l, first each center point $p^f_{h,l}$ of a face f of a hexagonal base H_h is transformed to the polar parameter space and then $\bar{C}^f_{h,l}$ is added to the corresponding bin (ρ, θ) of the accumulator array.

For the map of Fig. 6.2(a) the accumulator of the HT (Fig. 6.2(b)) has a maximum score at $\rho = 0$ m and $\theta = 46.5$ deg (and $\theta = 226.5$ deg), which is expected after visual inspection.

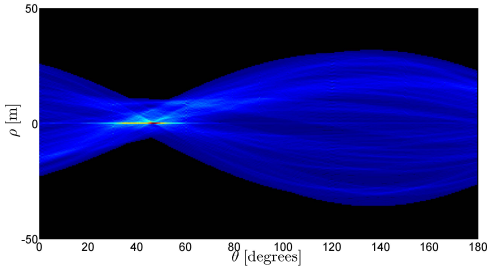
For layer L_l the HS is calculated as follows:

$$\mathrm{HS}(\theta_j, l) = \sum_{i=0}^{N_\rho - 1} \mathrm{HT}(\theta_j, \rho_i, l)^2, \quad 0 \leq l \leq N_z - 1, \tag{6.6}$$

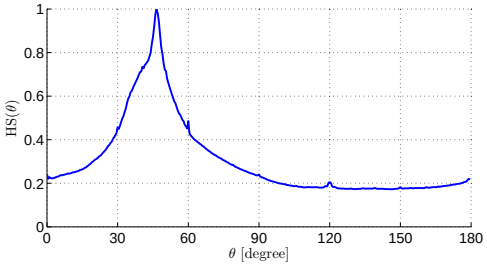
For each layer, the local maxima indicate a higher probability of a strong angular preference at the corresponding θ value. For the single-layered map of Fig. 6.2(a) the HS (Fig. 6.2(c)) peaks at $\theta = 46.5$ deg (and $\theta = 226.5$ deg), which is independent of ρ and thus independent of (x, y).



(a) FootSLAM posterior map of an office environment. The red dot corresponds to an arbitrarily chosen center of the coordinate system to compute the HT. The dashed line represents the perpendicular line to the main linear motion contributions (corresponding to walking along the corridor) learned by the HT. The solutions $\{\theta = 46.5\,\text{deg}, \theta = 226.5\,\text{deg}\}$) represent the angle between the normal and the x axis (the y axis grows downwards).



(b) Representation of the accumulator array of the HT of the FootSLAM map of Fig. 6.2(a), with period 180 deg: A jet color scale encodes the value of the accumulator array. The maximum is located at $\rho = 0$ m and $\theta = \{46.5, 226.5\}$ deg.



(c) Normalized Hough Spectrum of the FootSLAM map of Fig. 6.2(a), with period 180 deg. The maximum is located at $\theta = \{46.5, 226.5\}$ deg.

Figure 6.2: Example of a FootSLAM posterior map 6.2(a), its corresponding HT accumulator array 6.2(b) and the resulting angular spectrum based on the HS 6.2(c).

6.3.2 Spatial Spectra of FootSLAM Maps along the Cartesian Axes

The spatial spectra of a FootSLAM map shall reflect the main features of a map along the Cartesian axes.

To compute the spatial spectra, for each visited hexagonal prism $\{h, l\}$, we project the

count ratio of faces $\mathcal{F}_x = \{0, 1, \ldots, 5\}$ onto the x coordinate of the center of the hexagonal base and the count ratio of faces $\mathcal{F}_y = \{1, 2, 4, 5\}$ onto the y coordinate of the center of the hexagonal base (Fig. 6.3).

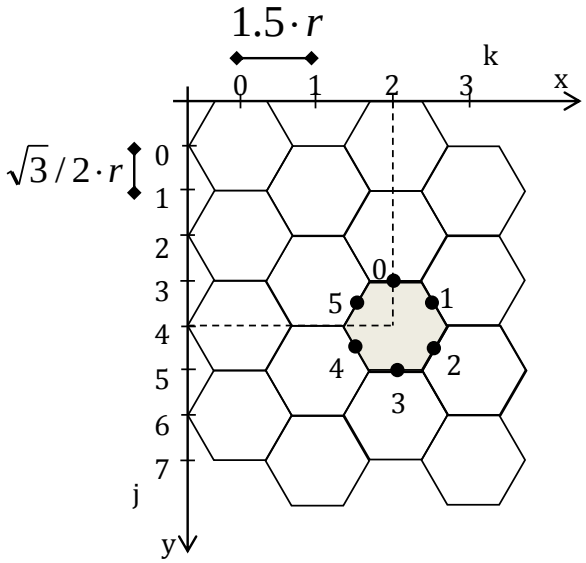


Figure 6.3: Top view of the hexagonal grid in Fig. 4.1. The indices k and j correspond to the discretization used for the projection of a FootSLAM map onto the x and y axes respectively. For example, the hexagon depicted in gray (column $k = 2$, row $i = 2$) shall be projected onto $k = 2$ and $j = 4$. We have marked with a dot the center of each face to project and its associated face number f.

X Spatial Spectrum

We can compute the spatial spectrum along the x axis (XS) of a layer L_l of hexagonal prisms as follows:

$$\begin{aligned} \mathrm{XS}(k,l) = \\ \sum_{i=0}^{N_y-1} \left[\bar{C}^0_{h(k,i),l} + \sum_{f=1, f\neq 3}^{f=4} \bar{C}^f_{h(k,i),l} \cdot \cos(\pi/3) + \bar{C}^3_{h(k,i),l} \right], \\ 0 \leq k \leq N_x - 1, \\ 0 \leq l \leq N_z - 1, \end{aligned} \tag{6.7}$$

where $h(k, i)$ refers to the index of the hexagonal base located at column k and row i of the prism grid (Fig. 6.3).

Y Spatial Spectrum

To obtain the spatial spectrum along the y axis (YS) of a given layer L_l of hexagonal prisms, for each row of hexagonal bases i, $(0 \leq i \leq N_y - 1)$, there are two y coordinates onto which we can project the count ratio, one for even columns and one for odd columns. We call j the "wide-sense" row number (Fig. 6.3):

$$\begin{aligned} \mathrm{YS}(j,l) = \sum_{k=0}^{N_x-1} \sum_{f \in \mathcal{F}_y} \bar{C}^f_{h(k,i),l} \cdot \sin(\pi/3), \\ \text{with } j = i \cdot 2 + i \mod 2, \\ 0 \leq i \leq N_y - 1, \\ 0 \leq l \leq N_z - 1. \end{aligned} \tag{6.8}$$

6.3.3 Combining Maps: Particle's Map vs. Posterior Map

Ozkucur and Akin [116] use posterior maps to find the geometric transformation between two maps. Then, when a robot transforms the posterior map of another robot, the transformed map is merged with each one of the particle's maps. To find the transformation between two maps, Dinnissen [39] uses the map estimate of the particle with the highest weight of each robot's particle filter.

Although our proposed approach can be applied on a per particle's map basis, we have chosen to work with posterior maps (as defined in (2.63)) because they encode all the particles' map hypotheses, it scales well with the number of particles and because of the reliable results obtained by [136]. From now on, we will refer to FootSLAM posterior maps as FootSLAM maps, but notice that the following algorithms can be applied to a particle's map.

6.4 Geometric Transformation Between Two Maps based on Angular and Spatial Spectra

Given two FootSLAM maps, we will use M^P to refer the FootSLAM map with point set $\mathcal{P}$ that is the fixed reference and M^Q to refer to the FootSLAM map with point set $\mathcal{Q}$ that we transform to fit M^P. We define $\mathcal{Q}^R = T(\mathcal{Q}, \Delta\theta, 0.0, 0.0)$ as the rotated point set and $\mathcal{Q}^S = T(\mathcal{Q}^R, 0.0, \Delta x, \Delta y)$ as the shifted rotated point set.

To estimate the transformation between $\mathcal{P}$ and $\mathcal{Q}$ (or M^P and M^Q), we follow the algorithm outlined in [28] but applied to FootSLAM maps instead of images:

1. Pre-align the angular features of M^P to the Cartesian axes by computing the HS and applying the rotation value $\Delta\theta$ with the maximum score. For simplicity, M^P represents this already aligned map.
2. Cross-correlate the Hough spectra of M^P and M^Q to find the angular displacement $\Delta\theta$ between them.
3. Rotate M^Q by $\Delta\theta$ to obtain M^{Q^R}.
4. Cross-correlate the spatial spectra of M^P and M^{Q^R} to find the spatial displacement between them, Δx and Δy.
5. Apply Δx and Δy to M^{Q^R} to obtain M^{Q^S}.
6. Combine M^{Q^S} with M^P.

In our extension of FootSLAM to multistory buildings, the transformation search problem remains the same if we assume that the relative position of the maps along the z axis is known (for example when the pedestrians carry a barometer that provides absolute height).

6.4.1 Projection of Multistory FootSLAM Maps

As explained in Section 2.6.2, applying a transformation to a FootSLAM map is always done in conjunction with a projection onto a target grid, i.e. a reference grid for all maps we wish to combine. This is a prerequisite to be able to apply a correlation function between two maps or add their contributions.

The projection is performed with the help of two factors, namely a distance factor and an angular factor. These two factors are used to compute the proportion of the transition counts of an edge of a hexagonal base that shall be assigned to any edge in the target grid.

In this section we propose an alternative to the distance factor presented in [136] and that also accounts for the upper and lower face transition counts of a multistory map. The angular factor is explained in Section 3.2 of [136] and is only applied to the lateral faces.

New Distance Factor to Project Lateral Faces

The distance factor proposed in [136] is computed as the distance between the center of each transformed hexagon and the center of each target hexagon that lies in the vicinity of the transformed hexagon. The proportion of the counts assigned to each target hexagon is defined by a *weight* that is inversely proportional to the distance. Such a weight is not robust against distances close to zero and different scaling factors between the transformed and target prisms.

Here we propose a new distance factor for the lateral faces of a prism. Given a transformed prism of M^{Q^S} (drawn with dashed lines in Fig. 6.4(a))), for each lateral face we can draw a circle with the diameter set to the radius of the prism's hexagonal base (red circle in Fig. 6.4(a)). The distance factor is defined as the overlapping area between that circle and the circles corresponding to the lateral faces in the target grid (blue circles in Fig. 6.4(a)).

Now, for each target lateral face (marked with the blue circles) we multiply the weight based on the angular factor and this distance factor and normalize them to sum to 1. The final weight represents the proportion of the transformed face's transition counts that will be assigned to each target face.

Distance Factor for Upper and Lower Faces

To project the transition counts of the upper and lower faces of a transformed prism we can draw a circle centered at the center of its hexagonal base and with the same radius (red circle in Fig. 6.4(b)). The distance factor is defined as the overlapping area between that circle and the circles corresponding to the hexagonal bases in the target grid (blue circles in Fig. 6.4(b)). The proportion of the transition counts of the upper and lower face that is assigned to each target prism is the distance factor after being normalized to sum to one over all target prisms (marked in blue).

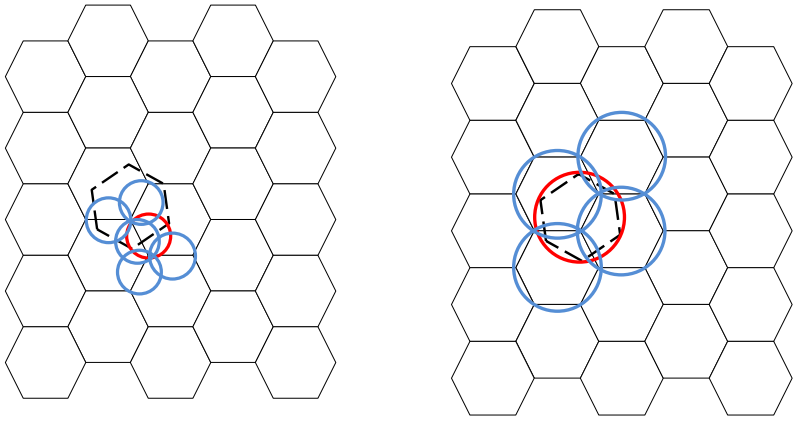
6.4.2 Cross-correlating Two Angular or Spatial Spectra

The cross-correlation between two angular or spatial spectra measures the similarity between the two spectra as a function of an angular or spatial displacement, respectively. Thus, local maxima in the cross-correlation may provide a list of possible candidates for the angular and spatial displacements to apply to a map to best explain the data of the other map.

Given the angular spectra at layer L_l of two point sets, $\mathrm{HS}_{\mathcal{P}}(\theta_j, l)$ and $\mathrm{HS}_{\mathcal{Q}}(\theta_j, l)$ respectively, we can compute the angular cross-correlation (AC) between them as their circular cross-correlation, with bin size $\Delta_\theta^q = \pi/N_\theta$ and period π:

$$\mathrm{AC}_{\mathcal{P},\mathcal{Q}}(\Delta\theta, l) = \sum_{j=0}^{j=N_\theta-1} \mathrm{HS}_{\mathcal{P}}(\theta_j, l)\mathrm{HS}_{\mathcal{Q}}(\theta_j + \Delta\theta, l), \quad 0 \leq \Delta\theta \leq (N_\theta - 1) \cdot \Delta_\theta^q. \tag{6.9}$$

On the other hand, given the spatial spectra at layer L_l over the x axis of two maps



(a) Representation of the computation of the distance factor for one lateral face as the overlapping area between the red circle (corresponding to the a lateral face of one prism of the transformed map M^T) and each blue circle (corresponding to any lateral face of a prism of the target grid with which the red circle overlaps).

(b) Representation of the computation of the distance factor for the upper and lower faces as the overlapping area between the red circle (corresponding to the upper or lower face of a prism of the transformed map M^T) and each blue circle (corresponding to the upper or lower face of a prism of the target grid with which the red circle overlaps).

Figure 6.4: Illustration of the computation of the distance factors. The hexagon in dashed lines is the top view of the prism we are projecting and the hexagons in continuous lines represent the top view of the target grid.

$\mathrm{XS}_{\mathcal{P}}(k,l)$ and $\mathrm{XS}_{\mathcal{Q}}(k,l)$, we can compute their cross-correlation as follows:

$$\mathrm{XC}_{\mathcal{P},\mathcal{Q}}(\Delta k, l) = \sum_{k=-\infty}^{k=\infty} \mathrm{XS}_{\mathcal{P}}(k,l) \cdot \mathrm{XS}_{\mathcal{Q}}(k+\Delta k, l), \quad -\infty \leq \Delta k \leq \infty. \tag{6.10}$$

Likewise, we can compute the cross-correlation over the y axis at layer L_l by applying:

$$\mathrm{YC}_{\mathcal{P},\mathcal{Q}}(\Delta j, l) = \sum_{j=-\infty}^{j=\infty} \mathrm{YS}_{\mathcal{P}}(j,l) \cdot \mathrm{YS}_{\mathcal{Q}}(j+\Delta j, l), \quad -\infty \leq \Delta j \leq \infty. \tag{6.11}$$

Finally, to obtain the translation in meters $(\Delta x, \Delta y)$ (see Fig. 6.3):

$$\Delta x = 1.5 \cdot r \cdot \Delta k \text{ [m]}, \tag{6.12}$$

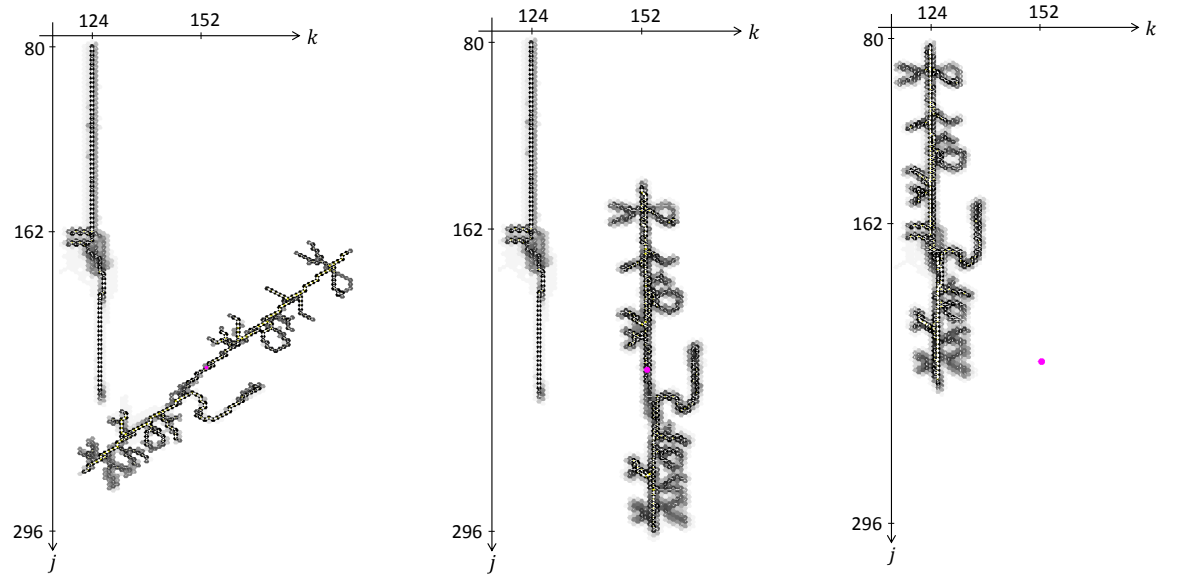
$$\Delta y = \sqrt{3}/2 \cdot r \cdot \Delta j \text{ [m]}. \tag{6.13}$$

In addition, we can compute the cumulative cross-correlation over all prism layers L_l for any of the cross-correlations defined above by summing the cross-correlation over all layers weighted by the geometric average of the total number of counts for that layer $C_l = \sum_h C_{h,l}$ between the two maps. For example, for AC (6.9):

$$\mathrm{AC}_{\mathcal{P},\mathcal{Q}}(\Delta\theta) = \sum_{l=0}^{l=N_z-1} \mathrm{AC}_{\mathcal{P},\mathcal{Q}}(\Delta\theta, l) \cdot \sqrt{C_l^{\mathcal{P}} \cdot C_l^{\mathcal{Q}}}. \tag{6.14}$$

6.4.3 Map Combination Example

In this section we show how to combine two FootSLAM maps of the same environment, in particular the maps shown in Fig. 6.5(a).



(a) M^P (left) and M^Q (right).

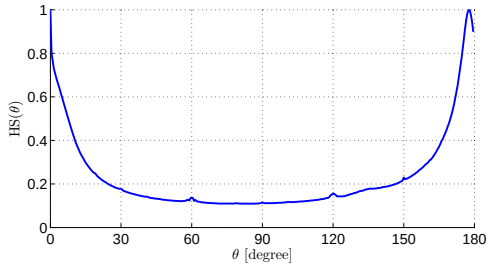
(b) M^P (left) and M^{Q^R} (right).

(c) Combined FootSLAM map.

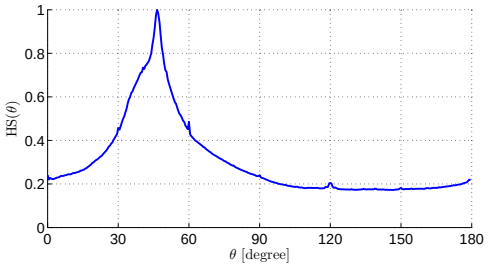
Figure 6.5: Two FootSLAM maps during the process of being combined.

The first step is to find the angular displacement between the two maps. Given the HS of each map (Figures 6.6(a) and 6.6(b)), we can compute the angular cross-correlation using (6.9) (Fig. 6.6(c)). The maxima are located at $\Delta\theta = 132.5\,\mathrm{deg}$ and $\Delta\theta = 312.5\,\mathrm{deg}$. In this case, the correct angular displacement is $\Delta\theta = 312.5\,\mathrm{deg}$. Shortly we will show how to discriminate between these two solutions.

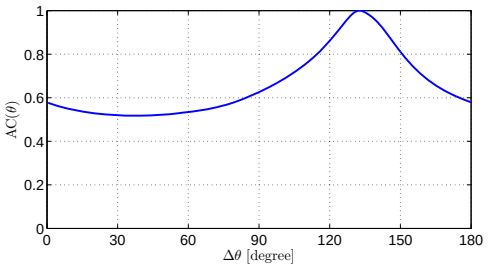
We apply a rotation of $\Delta\theta = 312.5\,\mathrm{deg}$ to M^Q to obtain M^{Q^R} with $\mathcal{Q}^{\mathcal{R}} = T(\mathcal{Q}, 312.5\,\mathrm{deg}, 0.0, 0.0)$, depicted on the right side of Fig. 6.5(b). The maps are clearly still shifted with respect to each other.



(a) Normalized Hough Spectrum $\mathrm{HS}_{\mathcal{P}}$ of the map on the left side of Fig. 6.5(a), with period 180 deg.



(b) Normalized Hough Spectrum $\mathrm{HS}_{\mathcal{Q}}$ of the map on the right side in Fig. 6.5(a), with period 180 deg.



(c) Normalized angular cross-correlation $\mathrm{AC}_{\mathcal{P},\mathcal{Q}}(\Delta\theta)$, with period 180 deg. The maximum is located at $\Delta\theta = 132.5\,\mathrm{deg}$ (and $\Delta\theta = 312.5\,\mathrm{deg}$)

Figure 6.6: Normalized Hough Spectra of maps in Fig. 6.5(a) (Fig. 6.6(a) and Fig. 6.6(b)) and angular cross-correlation between them (Fig. 6.6(c)).

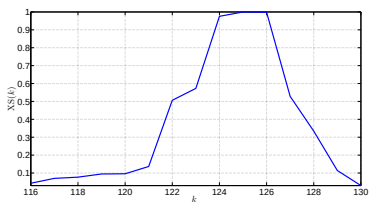
The next step is to obtain the spatial spectra of both maps and compute their cross-correlations to obtain the translation between them (Fig. 6.7). Remember that we have pre-aligned the angular contributions of M^P with the x and y axes so that the spatial spectra effectively reflect the contributions of the maps along the Cartesian axes.

The maxima of the spatial cross-correlations along x and y are located at $\Delta k = -29$ and $\Delta j = -62$, which correspond to $\Delta x = -21.75$ m and $\Delta y = -26.85$ m, respectively. Fig. 6.5(c) shows the resulting combined map, i.e. $M^C = M^P + M^{\mathcal{Q}^S}$ with $\mathcal{Q}^S = T(\mathcal{Q}^R, 0.0, -21.75\ m, -26.85\ m)$.

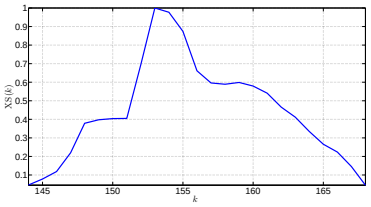
6.4.4 Using Lists of Possible Values

When merging maps, we need to identify regions that overlap between them. For example, Birk and Carpin [19] propose using a function that measures how much two occupancy grid maps overlap and searches stochastically for the best transformation.

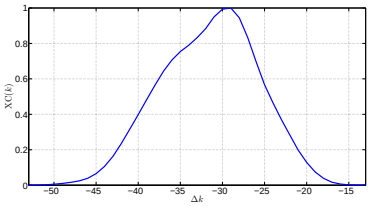
To quantify how well the data of one map explains the other, Section 3.3 of [136] chooses a likelihood function rooted in the Bayesian formulation of FootSLAM with an extra



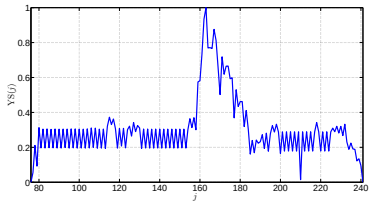
(a) Normalized X spectrum of $\mathcal{P}$, $XS_{\mathcal{P}}$, depicted on the left side of Fig. 6.5(b).



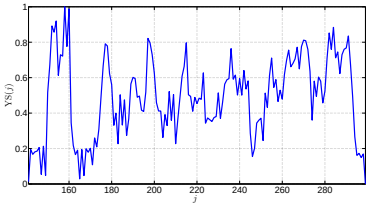
(b) Normalized X spectrum of $\mathcal{Q}^R$, $XS_{\mathcal{Q}^R}$, depicted on the right side of Fig. 6.5(b).



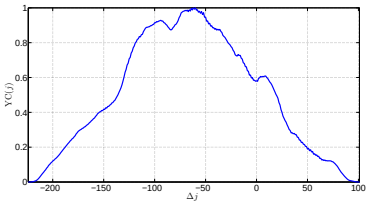
(c) Normalized spatial cross-correlation along the x axis, $XC_{\mathcal{P},\mathcal{Q}^R}$. The maximum is located at $\Delta k = -29$



(d) Normalized Y spectrum of $\mathcal{P}$, $YS_{\mathcal{P}}$, depicted on the left side of Fig. 6.5(b).



(e) Normalized Y spectrum of $\mathcal{Q}^R$, $XS_{\mathcal{Q}^R}$, depicted on the right side of Fig. 6.5(b).



(f) Normalized spatial cross-correlation along the y axis, $YC_{\mathcal{P},\mathcal{Q}^R}$. The maximum is located at $\Delta j = -62$.

Figure 6.7: Normalized spatial spectra and cross-correlations of the maps depicted in Fig. 6.5(b). Index k represents the column number in the prism grid and index j represents the "wide-sense" row number, which considers two rows per prism grid row (Fig. 6.3).

heuristic term (2.71). A map that represents where one person has walked should be a good fit in explaining the walk of another in the same environment.

In this contribution we extend the logarithmic likelihood value (logLV) of (2.71) to accommodate the transitions across the upper and lower faces, yielding:

$$\begin{aligned} &\log\mathrm{LV}(M^P, M^Q) = \\ &= \frac{\displaystyle\sum_{\{h,l\}\in\mathcal{H}_{MQ}} \sum_{f=0}^{f=7} C_{h,l}^{fQ} \cdot \log\left(\frac{C_{h,l}^{fP} + \alpha_{h,l}^{f}}{C_{h,l}^{TP} + \alpha_{h,l}^{T}}\right)}{\displaystyle\sum_{\{h,l\}\in\mathcal{H}_{MQ}} C_{h,l}^{TQ}} \\ &+ \frac{\beta \cdot \displaystyle\sum_{\{h,l\}\in\mathcal{H}_{MQ}} C_{h,l}^{TQ} \cdot C_{h,l}^{TP}}{\displaystyle\sum_{\{h,l\}\in\mathcal{H}_{MQ}} C_{h,l}^{TQ}} \end{aligned} \tag{6.15}$$

To be able to track multiple hypothesis in a non-iterative fashion, we keep $N_{\max}$ hy-

potheses for the angular displacement (the highest $N_{\max}$ local maxima of the angular cross-correlation between two maps). For each angular displacement candidate, we compute XC and YC and transform $M_{\mathcal{Q}^{\mathcal{R}}}$ with all possible spatial combinations of the $N_{\max}$ greatest local maxima of XC and the $N_{\max}$ greatest local maxima of YC. Thus, we track a total of $N_{\max}^3$ possible geometric transformations. Finally, we decide which transformation to apply by selecting the one that yields the maximum logLV (6.15).

Operating like this we can overcome possible rotational or spatial symmetries of the building as well as building structures that are not orthogonal. In addition, tracking a number of transformation hypothesis helps confront the problem of merging maps that do not overlap in great degree.

6.4.5 Transformation Search Range

In this work we assume some knowledge regarding the relative starting position and heading of the pedestrians. In the case of the angular displacement, we use this information to discriminate between the solutions $\Delta\theta$ deg and $\Delta\theta + 180$ deg.

In the professional market domain, this is a feasible assumption, since most of the walks start outside around the same coordinates (for example, at the firetruck) and the team members enter the building using only a few entrances. Alternatively, GNSS or other pre-installed infrastructure might be partially available during the walks [131][26].

6.5 Computational Complexity Analysis

In this section we analyze the computational complexity of the map combination step based on angular and spatial spectra and compare it to the brute-force approach in [136], which is the only existing implementation. We are interested in studying the complexity growth of the algorithm as the space visited by the pedestrians grows asymptotically to infinity. For this purpose, we will rely on asymptotic notation to express the worst-case scenario (Section 2.1).

6.5.1 Complexity of the Brute-force Approach

As we stated in Section 2.6.2, the map combination step in [136] applies N_x^{tr} translation values along the x axis times N_y^{tr} translation values along the y axis times N_r rotations values to $\mathcal{Q}$ and relies on the Bayesian-motivated function in (6.15) to determine the best fit to $\mathcal{P}$.

The computational complexity of cross-correlating two multistory maps using (6.15) grows in the number of prisms of the maps $\mathcal{O}(N_{\mathcal{H}}) = \mathcal{O}(N_x \cdot N_y \cdot N_z)$. This operation is performed $N_r \cdot N_x^{\text{tr}} \cdot N_y^{\text{tr}}$ times, which is of the order of $\mathcal{O}(N_x \cdot N_y) = \mathcal{O}(N_H)$ since we can consider N_r

independent of the area of the maps, i.e. $N_r = O(1)$. Thus, the total complexity order is $\mathcal{O}(N_{\mathcal{H}} \cdot N_H) = \mathcal{O}(N_z \cdot N_H^2)$.

6.5.2 Complexity of the Proposed Approach

For our proposed algorithm, the complexity study is based on the following premises:

1. The input space for the HT comprises, for each prism layer with 6 lateral faces, $N_s = 6 \cdot N_H = \mathcal{O}(N_H)$ points.
2. The number of bins N_θ is independent of the area $N_\theta = \mathcal{O}(1)$ but the number of bins N_ρ depends linearly on the largest dimension of one layer in the hexagonal prism grid $N_\rho = \mathcal{O}(N_x)$ (for simplicity we assume $N_x = N_y$).

In the following, we study the complexity of each one of the parts of our HT-based algorithm for one single prism layer. The resulting complexity for multistory maps is computed by integrating the complexity per layer for all layers.

Complexity of HT Algorithm

When calculating the HT we take the N_s points of the input space and transform them into a $N_\rho \cdot N_\theta$ Hough space. This requires computational complexity in $\mathcal{O}(N_s) = \mathcal{O}(N_H)$.

Complexity of HS Algorithm

Computing the HS requires, for each bin $0 \leq j \leq N_\theta - 1$, to compute the sum of N_ρ elements. The computational complexity of the HS grows with $\mathcal{O}(N_\theta N_\rho) = \mathcal{O}(N_x)$.

Complexity of Spatial Spectra Computation

The computation of the spatial spectra along the x axis requires, for each column $0 \leq k \leq N_x - 1$ to compute the sum over the N_y rows of 6 elements, i.e. it has a complexity growth of $\mathcal{O}(N_x \cdot N_y) = \mathcal{O}(N_H)$. The same reasoning applies to the computation of the spatial spectra along the y axis.

Complexity of the Cross-correlations

The cross-correlation of two HSs requires $\mathcal{O}(N_\theta \cdot N_\theta) = O(1)$; the cross-correlation of two spatial spectra requires $\mathcal{O}(N_x \cdot N_x) = \mathcal{O}(N_H)$.

This is the main reduction in complexity gained by the proposed algorithm. There is no longer a quadratic dependency with the explored area: it has been reduced to linear.

Complexity for All Prism Layers

The complexity of the spectra-based approach to finding the geometric transformation between two maps is, after integrating for all layers:

$$\mathcal{O}(N_z \cdot N_H) = \mathcal{O}(N_{\mathcal{H}}).$$

Remember that we track $N_{max}^3 = \mathcal{O}(1)$ candidates for the transformation (Section 6.4.4) and use (6.15) to decide which transformation to apply. As explained in Section 6.5.1, applying (6.15) has a computational complexity of $\mathcal{O}(N_{\mathcal{H}})$.

Thus, the complexity of the proposed approach is $\mathcal{O}(N_z \cdot N_H) = \mathcal{O}(N_{\mathcal{H}})$. We have been able to *reduce the complexity of the map merging step from quadratic to linear in the explored area of a floor.*

6.6 Experimental Validation

6.6.1 Experimental Settings

In all the experiments a pedestrian walked with foot-mounted IMU, in particular the cable connected MTx from x-Sens and the measurements were processed by the UKF in [166] to generate step measurements. The pedestrians deliberately revisited areas, i.e. closing the loop, and walked at a normal pace. The datasets were processed in an offline fashion by doing the following:

1. Process each dataset with FootSLAM to build individual maps.
2. Order the maps by increasing map entropy as defined in (3.10).
3. Combine the two maps with the lowest map entropy into an individual map using the angular and spatial spectra of the maps as explained in this chapter.
4. Go back to 2. and repeat until only one single combined map is left.

The common settings for all walks are: $N_p = 10\ 000$, $N_x = 256$, $N_y = 256$, $N_\theta = 360$ deg (i.e. $\Delta_\theta^q = 0.5\,\text{deg}$) $\Delta_\rho^q = 0.5$ m, $r = 0.5$ m.

Experiment 1

A pedestrian undertook 3 walks within the first floor of a typical office environment in 2010/2011 (DLR building TE01 in Oberpfaffenhofen, Germany). The building is structured into a rectangular corridor with offices on both sides. Fig. 6.8 shows the individual maps (on the top) and their combination (on the bottom).

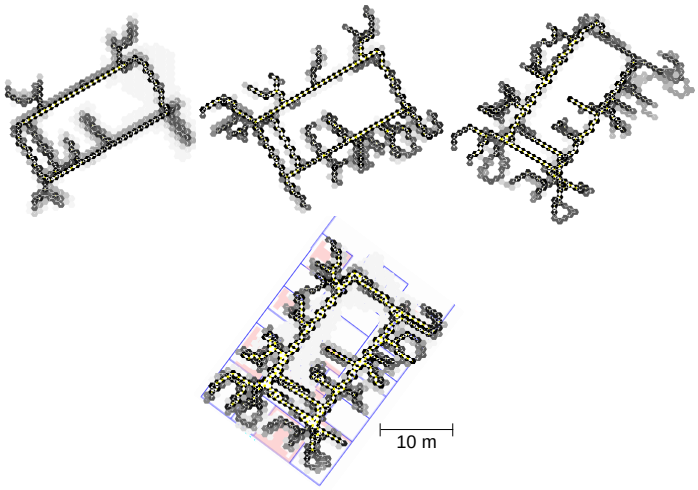


Figure 6.8: Three FootSLAM maps of DLR building TE01 (on the top) and their combined map. The building layout (in blue) with the furniture layout (in pink) have been manually aligned to the FootSLAM map and are shown only as a reference.

Experiment 2

Two pedestrians undertook 6 walks within the second floor of an office building (DLR building 103 in Oberpfaffenhofen, Germany) over several months in 2012. The structure comprises two long corridors separated by a common area. The pedestrians visited almost all offices on both sides of the corridors along with the two balconies. Fig. 6.9 shows the individual maps and their combination.

Experiment 3

One pedestrian collected four datasets on the second floor of the Stata Center in Cambridge, Massachusetts (USA) in March 2011. The building structure is far from a typical office environment, with almost no rectilinear corridors or rectangular rooms. Fig. 6.10 shows the individual maps and their combination. In this case, the use of a list of local maxima (Section 6.4.4) turned crucial for finding the correct transformation.

Experiment 4

A pedestrian collected two datasets within six floors of DLR Building 103 in September 2012. The pedestrian walked within each floor and used stairways to transition from one floor to the next, i.e. no elevators or escalators were used. The floor separation in the building is 3.54 m and we used $N_z = 64$ and $H_{\mathcal{H}} = 0.59$ m. Fig. 6.11 shows the two multistory maps and the resulting combined map. To illustrate how the HT

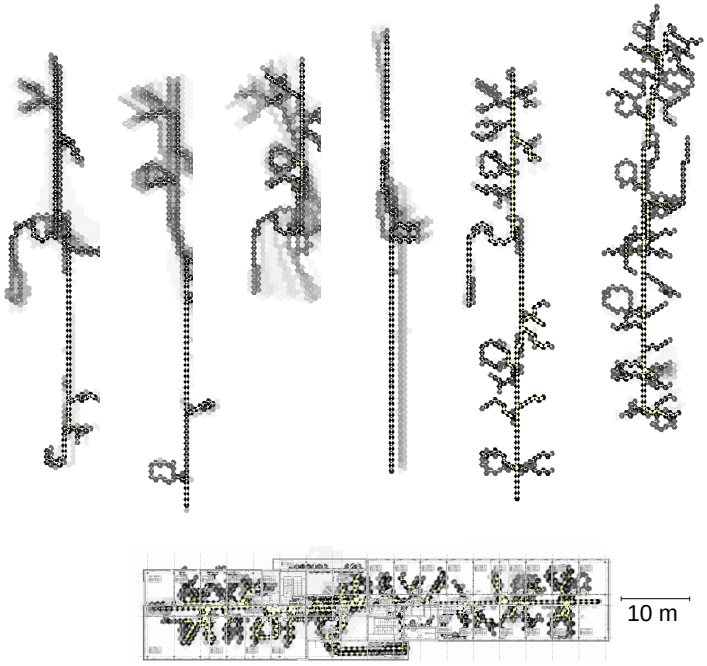


Figure 6.9: Six FootSLAM maps of DLR building 103 (on the top) and their combined map. The building layout has been manually aligned to the FootSLAM map and is shown only as a reference.

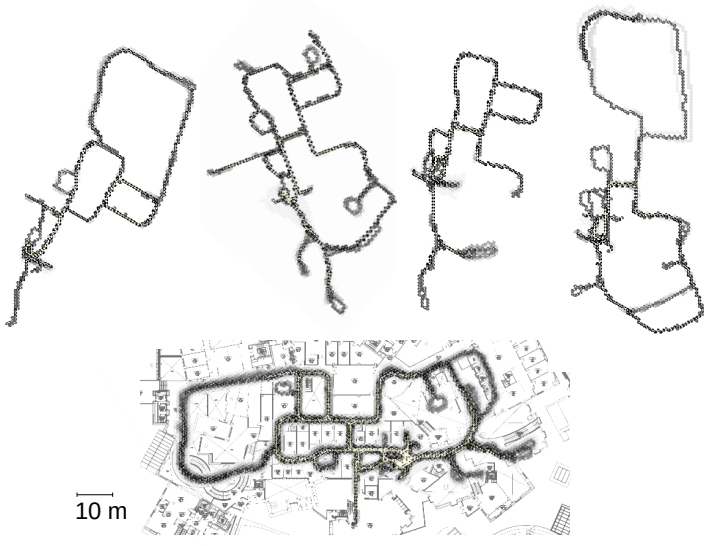


Figure 6.10: Four individual maps of the second floor of the Stata Center (on the top) and their combined map (on the bottom). The building layout has been manually aligned to the FootSLAM map and is shown only as a reference.

works for maps with multiple layers Fig. 6.13 shows the angular spectra and angular cross-correlation of the maps depicted in Fig. 6.11.

6.6.2 Results

The results suggest that we can successfully apply the HT to reduce the computational and time requirements of the map combination step of FeetSLAM for typical buildings with strong angular preferences that constrain motion to be approximately linear. In particular, we found that HT-based FeetSLAM yields the same qualitative results as the only FeetSLAM implementation to date [136] for all the data collected. The map combination process has been sped up and the time requirements to combine two maps have been reduced from tens of minutes to an average of 4 seconds for each pair of maps.

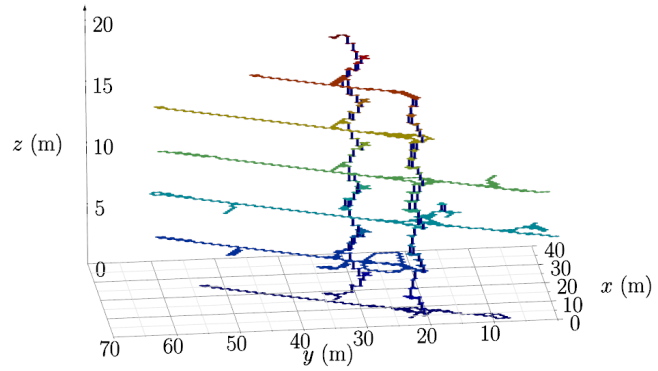
We have overlaid each floor with the available building plan and have manually checked that each visited office was correctly mapped, reflecting the presence of walls, furniture and other obstacles.

The resolution of the transformation search is limited by the number of bins N_θ and the radius of the hexagonal bases r (since we project a FootSLAM map onto the axes on a column and row basis). The values $N_\theta = 360$, i.e. $\Delta_\theta^q = 0.5\,\text{deg}$ and $r = 0.5$ m seem to yield successful results for all experiments.

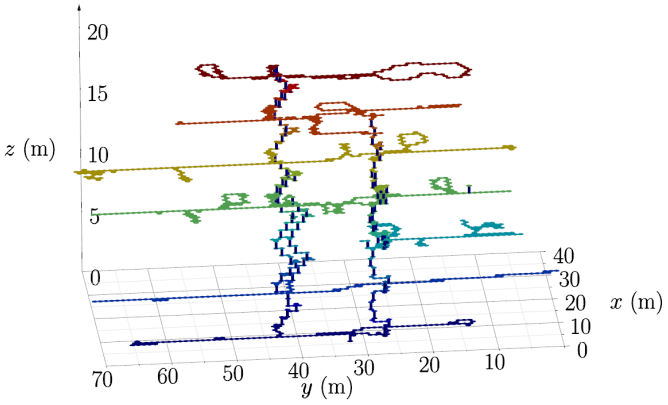
6.7 Discussion: Towards Online Map Merging

To quickly find the geometric transformation between two maps, we have applied the algorithm proposed by Carpin [28], which simplifies the transformation search using angular and spatial spectra, reducing the complexity of the map merging step from quadratic to linear in the area of a floor: Using the HT, we have been able to obtain the angular spectrum of a FootSLAM map; projecting the map onto the Cartesian axes, we have computed its spatial spectra. The novelty of our work lies in the fact that in our approach:

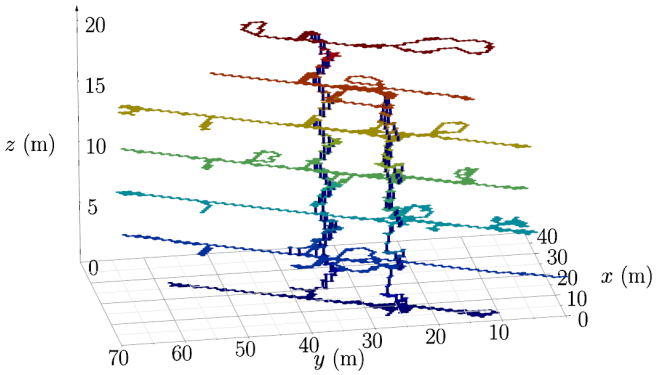
1. The mapping entities are walking pedestrians with free will instead of guided moving robots.
2. The pedestrians carry only inertial sensors, instead of laser scanner or cameras, which have lower data-rates, are low cost and raise fewer privacy issues.
3. The data are processed by a particle filter that generates a map composed of cells (hexagonal prisms) each associated with 8 probabilities, instead of an occupancy grid map with cells associated to a binary state.
4. The map nature is fundamentally different: instead of directly showing walls and obstacles. FootSLAM maps reflect human motion, which is constrained to be linear in typical buildings.



(a) Multistory map M^P.



(b) Multistory map M^Q.



(c) Combined map.

Figure 6.11: Two multistory maps M^P (Fig. 6.11(a)) and M^Q (Fig. 6.11(b)) and the resulting combined map (Fig. 6.11(c)). Different colors encode different floors. Hexagonal prisms with vertical transitions are marked with a blue rectangular polyhedron, reflecting the position of stairs.

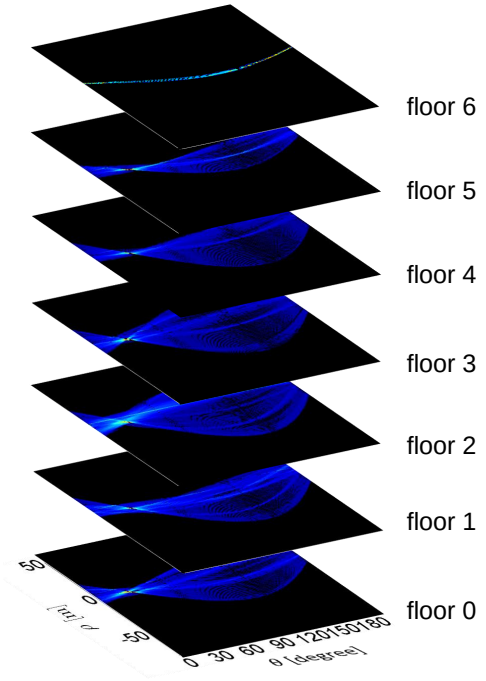


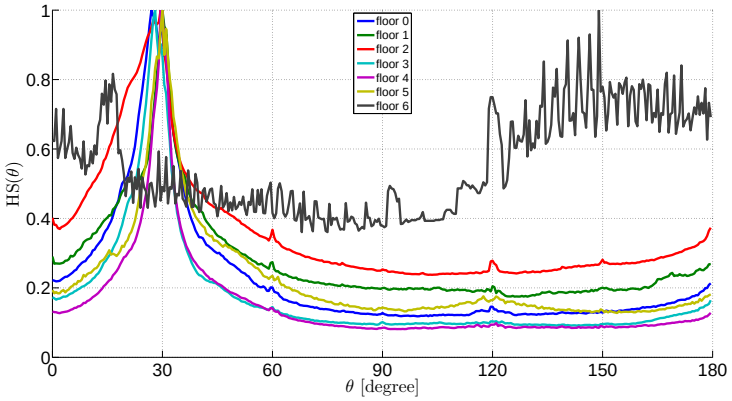
Figure 6.12: HT of each floor's motion map M^P in Fig. 6.11(a), each with period 180 deg.

5. We work directly with our map representation, instead of having to convert the map to a binary image.
6. We have applied the approach to multistory environments.
7. We have provided with a detailed analysis of the complexity of the proposed algorithm.

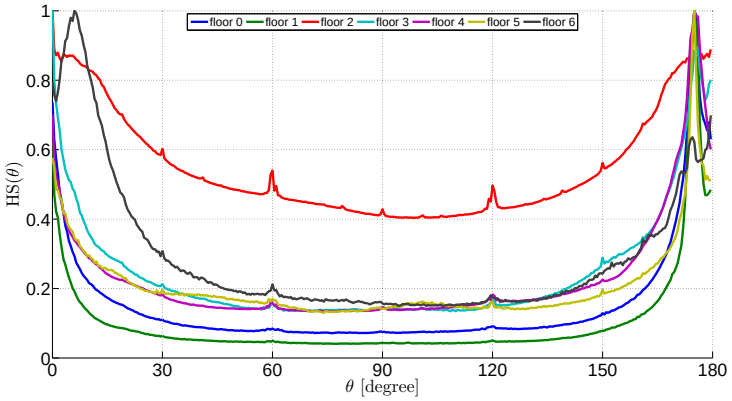
This reduction in complexity is fundamental for the generation of an indoor map database and may open the door to online applications, for example in security and emergency team operations (more in Chapter 7).

However, the limitations of the presented map merging algorithm with unknown starting pose and no rendezvous between the pedestrians are:

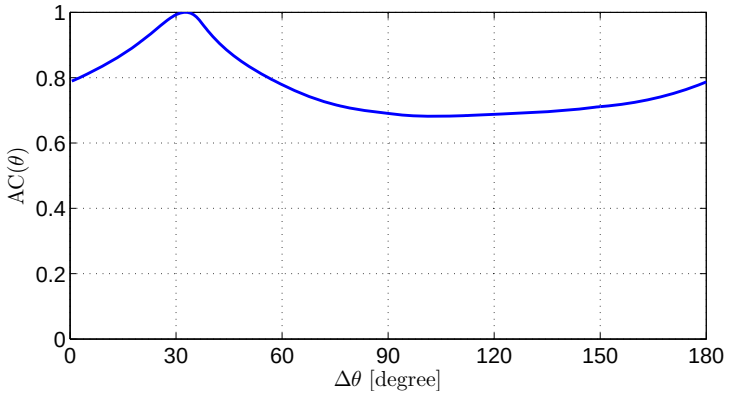
1. The maps must overlap in some degree. Knowledge of the starting pose of the pedestrians can help when the maps do not overlap in great degree. For example, due to the symmetries in the building, the third map on the top of Fig. 6.9 is successfully merged only when we roughly know the starting conditions of the walk.
2. The individual maps should have low uncertainty, i.e. the FootSLAM estimation process of the corresponding dataset must converge. This might require individual pedestrians to revisit areas in order to close the loop.
3. For robustness, we have chosen the HT as a basis to compute the angular spectrum of FootSLAM maps. Further work shall address using simpler line detection algorithms



(a) Normalized Hough Spectrum of each floor of the multistory map M^P in Fig. 6.11(a), with period 180 deg.



(b) Normalized Hough Spectrum of each floor of the multistory map M^Q in Fig. 6.11(b), with period 180 deg.



(c) Normalized angular cross-correlation $\mathrm{AC}_{\mathcal{P},\mathcal{Q}}(\Delta\theta)$, with period 180 deg.

Figure 6.13: Normalized Hough Spectra of the two multistory maps in Fig. 6.11(a) and Fig. 6.11(b) and their cross-correlation. Fig. 6.13(c) indicates that the maps are displaced 32.5 deg (correct solution) or $32.5 + 180 = 212.5$ deg.

that may perform with lower complexity.

4. Finding the angular displacement between two maps may fail in those cases in which the building shows angular symmetries or no angular preferences. Nevertheless, typical buildings exhibit clear angular preferences.

5. The map merging process based on the HT will fail if the maps do not share the same scale. A possible solution could be to augment FootSLAM maps with features [95], use the a shape-context metric [16] or identify key-points as in [158], making the algorithm scale invariant.

6. Our work assumes that the relative position along the z coordinates of two given maps is known. This may not always be the case. Without an external reference such as a barometer, this is a hard problem to resolve given that buildings are usually symmetric across floors. Another possibility is the include the local distortions of the magnetic field in the map [134]. Since the magnetic signature is different across floors, we expect this to allow floor levels to be differentiable to a much larger degree than the probabilistic map of human motion.

7. We have proposed one single alternative to the only existing implementation of [136], which shows a reduced computational complexity that seems to be sufficient for quasi real-time collaborative mapping. Other approaches should be explored in future work.

7 Large-scale Mapping and Applications

Chapter 6 has proposed a new algorithm to reduce the time requirements of combining maps in FeetSLAM. Our vision is to use such advances in FeetSLAM to generate a map database of the indoor world to allow the development of location-based applications. This chapter focuses on two main topics:

1. Estimating the time and resources necessary to build such a crowdsourced indoor map database. To do this, under a number of assumptions we will make estimates about the size of the indoor world, analyze the time to accurately map an area depending on how often it is visited and estimate the computational, memory and communication requirements to build this indoor map database.

2. Reviewing location-based applications, e.g. first responder applications vs. mass-market applications, and alternative deployment scenarios for FeetSLAM, e.g. with a centralized or decentralized approach to build or use crowdsourced maps.

We will show that using a crowdsourced approach we could collect sufficient step measurement data within a couple of months for those areas that are frequently visited (which usually coincide with those areas where location-based services are needed the most, e.g. a shopping mall). In addition, we will show that the computational effort to build an indoor map database of the whole world using FeetSLAM would require less than a month using 5000 cores and that the maps could be compressed and stored using only 200 gigabytes of memory.

The main contributions of this chapter have been published in [58] and in [57].

This chapter is organized as follows: in Section 7.1 we present the methodology used to estimate the requirements of large-scale mapping. Section 7.2 explains the characteristics of map-aided PDR. We continue by estimating the size of the indoor world in Section 7.3 and Section 7.4 introduces the concepts of visiting frequency and proportion of mapping individuals to compute the required time to map a given area. Next, Section 7.5 calculates the effort in terms of computational and memory requirements needed to map the indoor world and briefly addresses privacy issues. Finally, Section 7.6 outlines applications for human motion maps and possible deployment scenarios for FeetSLAM.

7.1 Methodology

We will base our work in this chapter on a number of assumptions:

- The majority of our society walks in some form or the other every day, or they conduct comparable forms of motion such as using a wheelchair.
- Some of these people will contribute to the mapping process. This could be as active volunteers or simply by accepting that data are collected anonymously during usage of a device or service.
- People are most likely to need navigation services in unfamiliar environments, usually outside private residences.
- The mapping process is largely governed by the normal usage of a building. In other words, we propose that motion actively directed by the goal of mapping will remain the exception.
- We postulate that any successful positioning scheme drawing on maps will need to be able to coast through unmapped or poorly mapped areas. (Map-aided PDR fulfills this requirement and will be a special case discussed below).
- Those places at which more people desire or use a navigation service correspond to places more frequented by people that can contribute to mapping.
- Much of our indoor world is heavily frequented by people, at least part of the time.

We have chosen in this chapter to approach the map generation problem from two directions. On the one hand we will look at small, individual areas of a building and look at the frequency with which they might be visited by humans. On the other hand we will look at the data that result directly from an average contributing person. By making assumptions about the proportion of actively contributing people within society we can make estimates of:

1. How long it will take to map different areas of a building, and
2. The computational resources required to achieve this globally.

This fits with a possible introduction scenario where the only factor that changes significantly over time is the proportion of pedestrians (ρ) who at any point in time might actively or passively contribute to a collaborative mapping effort. Coverage will grow most quickly where people tend to go, perhaps with a bias reflecting the social and technological background of the contributors (especially early adopters who might belong to specific social groups).

7.2 Map Characteristics of Map Aided PDR and FootSLAM

We recall that PDR drawing on maps for positioning is able to coast through unmapped or poorly mapped areas. Indeed, from a probabilistic perspective a FootSLAM map, in the absence of any observations, is equivalent in its mapping characteristics to an entirely open area that exerts no influence on human motion within it. Hence a PDR segment in an unmapped area will deteriorate in accuracy just as it would were a person to walk in a (known, i.e. mapped) nonrestrictive region. It is the strong motion restrictions imposed by walls that maintain the position accuracy while a pedestrian walks in a building; after entering a room through a door the accuracy might be on the order of 1 meter, deteriorating while the user is in a large space until the next constrictive opening is passed.

It is very likely that a typical building contains areas of varying visiting frequency, perhaps spanning several orders of magnitude. A pedestrian in such a building will usually enter through areas with (relatively) high visiting frequency and cover longer distances in areas of average to high visiting frequency. Similarly, the areas in the building that strongly channel pedestrians' motion are more likely to be visited more frequently in the first place (for example the main corridors in Figure 2.11). This advantageous situation will mean that during the course of FootSLAM mapping those areas that are most beneficial towards achieving good positioning accuracy are usually mapped more quickly than the less relevant areas.

7.3 How Big is Indoors?

With a world population of approximately 7 billion humans as of 2012, we will approximate the total number of pedestrians N_p we may have to localize to be $10 \cdot 10^9$.

In Europe (EU 27 plus Norway and Switzerland) there are approximately 512 million inhabitants spread over ca. 25 billion square meters [124], which means that there is a ratio of ca. 50 m^2 per person. If we assume that this number provides an estimate for the extent of indoor area per person, we can postulate that there are $S_w = 5 \cdot 10^{10}$ m^2 of indoor area in the world to be potentially mapped.

However, one must note that only 25% of the indoor areas in the EU correspond to non-residential areas, i.e. public/open spaces where users could strongly benefit from LBSs. Nevertheless, our goal here is to estimate the costs of mapping *all* kinds of indoor areas, without further differentiation of application domains.

7.4 Rate of Coverage

7.4.1 Walking Speed

Pedestrians exhibit different walking speeds depending on their gender, age and weather conditions among other factors. Speeds vary, on average, between 1.16 and 1.56m/s [94]. For example, using one single foot-mounted IMU to obtain a pedestrian's step measurements we can assume that the average step rate v is roughly one (double) step per second.

7.4.2 Visiting Frequency

We define the visiting frequency f of an area to be the number of people passing through it in a given period. We do not distinguish between the direction of travel or speed and we assume no pedestrian sources or drains. For the sake of simplicity and because of the discretization of current implementations of FootSLAM we will typically refer to an area of one square meter.

When quantifying the length of time needed to map indoor areas it is clear that we shall have to take into account the large differences in the number of times these areas are visited by people. But we have not found a source or process that could quantify the probability distribution of the visiting frequency in our very diverse indoor world. One might speculate that the visiting frequency distribution approximately follows a structure such as a Zipf distribution [167]. One approach to approximate it might be to model a typical day-in-my-life of a representative set of people and estimate where they spend their time (e.g. working in an office, going out, traveling, shopping, etc). To help understand and quantify the variability in visiting frequency, we shall define six profiles of visiting frequency of one square meter, in decreasing order of visiting frequency:

1. Maximal frequency: 2 people passing through the square meter per second: 172 800 visits per day (for example the area around a turnstile before entering a platform in an underground station of a big city).
2. High frequency: $\frac{0.1}{s}$ (one person every ten seconds): 8 640 visits per day (perhaps a busy museum or shop entrance, with roughly 3 million visitors per year).
3. Medium frequency: $\frac{0.01}{s}$ (one person every 100 seconds): 864 visits per day (area in front of a busy ATM; entrance to an elevator).
4. Low frequency: $\frac{0.001}{s}$ (one person every 1 000 seconds, i.e. just over 3 per hour): 86.4 visits per day (low frequented region of a typical office corridor).
5. Very low frequency: $\frac{0.0001}{s}$ (one person every 10 000 seconds, i.e. roughly ten persons per day): 8.64 visits per day (entrance of a residential building).

6. Minimal frequency: $\frac{0.00001}{s}$ (one person every 100 000 seconds, i.e. roughly one person per day): 8.64 visits in 10 days (a storage room that is not frequently visited, roughly once a day).

These profiles of visiting frequency will be later used to compute an estimate of the required time to map a given area.

7.4.3 Proportion of Mapping Individuals

Mobile phone penetration has reached almost 100% in modern society and it is to be expected that the proportion of user devices with high computing power resources will increase and eventually lead to a dominance of devices with a least smartphone computing capabilities. One can only speculate, however, with regards to the proportion of pedestrians ρ who at any point in time might actively or passively contribute to a collaborative mapping effort while going about their daily lives.

7.4.4 Time-to-map

FootSLAM as well as other many forms of Simultaneous Localization and Mapping will require several visits to a certain area in order to achieve a map with some degree of reliability or local coverage. As far as FootSLAM is concerned we can achieve a high accuracy of a local map after roughly 10 to 100 visits. Using these two limits we can use the above quantitative definitions and assumptions to compute the time-to-map of areas corresponding to each of the different profiles of visiting frequency:

$$T_{map} = \frac{N_{\text{vis}}}{f \cdot \rho}. \tag{7.1}$$

Figure 7.1 shows the time required to map different types of venues. The red rectangle shows the cases that are of particular interest and relevance. The required time to map an area ranges from 20 seconds for the turnstile type of area ($\rho = 0.25, f = 2\ \text{s}^{-1}, N_{\text{vis}} = 10$) to roughly one year for those areas that are less frequently visited, such as a residential entrance ($\rho = 0.0025, f = 10^{-4}\ \text{s}^{-1}, N_{\text{vis}} = 10$ and $\rho = 0.025, f = 10^{-4}\ \text{s}^{-1}, N_{\text{vis}} = 100$).

7.5 Effort

7.5.1 Computational Effort

FootSLAM

We will assume that a healthy and active person takes about $N_s = 10\ 000$ steps per day. We can also assume that 50% of this activity takes place in open/public spaces, with FootSLAM activated, i.e. $N_s^{\text{FS}} = 5\ 000$ steps.

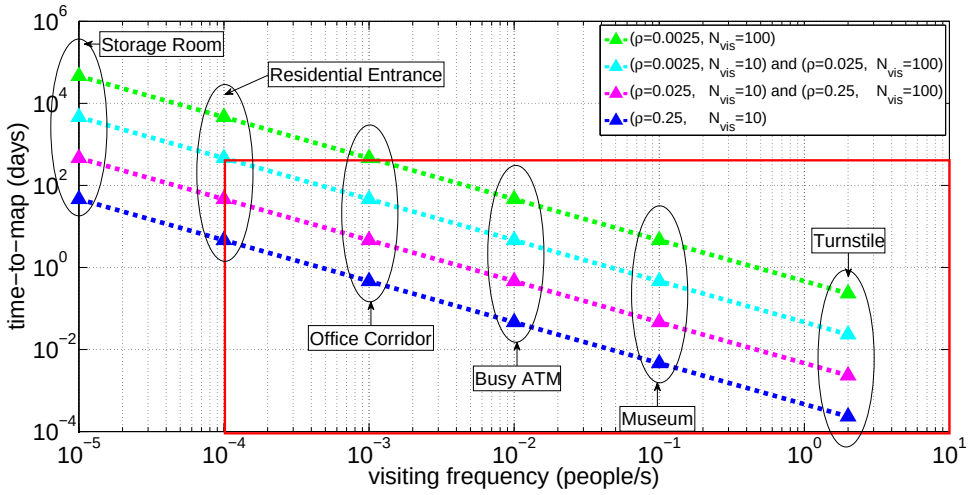


Figure 7.1: Time-to-map an area of one square meter given the frequency of visiting pedestrians f assuming different values for the density of collaborating pedestrians $\rho = \{0.0025, 0.025, 0.25\}$ and different needs of number of visits $N_{\text{vis}} = \{10, 100\}$. The red rectangle shows the cases that are relevant.

Thus, at a roughly walking rate v of one step per second the data collected during one day have a duration of $d = N_s^{\text{FS}}/v$ =5 000 seconds. If we process these data on a single core processor at 100 times faster-than-realtime rate, then the required CPU time to process the data of a single walk is: $T_{\text{CPU}} = d/100 = 50$ seconds.

FeetSLAM

Using the FeetSLAM algorithm the maps obtained by different walks within the same building can be merged to generate a more accurate and complete map of the indoor environment. For the following calculations, we will assume that an average building has a floor surface area of $S_b = 100$ m×100 m.

In case a large number of individual maps is available, they can be combined using a tree structure, whereby smaller groups of maps first combined. Later on, the combined maps of each group are merged until one single final map is available. If we choose to combine maps in groups of $N_m = 4$ maps and apply the map entropy metric (3.10) to select the order in which we combine the maps, then one iteration of the FeetSLAM algorithm takes about 3 minutes in a single core processor and the time to run 4 iterations - at which point we have shown good convergence results - will be bounded by 10 minutes (later iterations reduce the area where translations and rotation values are searched for and take less time). If we assume that $N_M = 64$ maps of an indoor environment are enough to achieve great accuracy (areas are revisited between 10 and 100 times) and that we can execute $N_g = N_M/N_m = 16$ FeetSLAM processes in parallel, we could obtain the total combined map of the building in $T_b = log_{N_m}(N_M) \cdot 10$ min$= 3 \cdot 10$ min $= 30$ min.

As a consequence, running FeetSLAM for our $S_w = 5 \cdot 10^{10}$ m^2 of estimated indoor area in groups of $S_b = 10^4$ m^2 and using N_c=5000 cores of standard desktop performance (ca.

3GHz) operating in parallel, we could have the *entire* indoor world mapped in $T_w = S_w/S_b \cdot T_b/N_c \approx 21$ days, i.e. in less than a month.

If we can apply the Hough spectrum and spatial spectra (Chapter 6) to quickly find the geometric transformation between two maps, combining $N_m = 4$ maps takes of the order of 10 seconds, and we could reduce even further the amount of time needed to map indoor environments.

7.5.2 Memory Requirements

Uncompressed, a simple multistory FootSLAM map using hexagonal prisms requires 8 counters per hexagon prism. Assuming that the prisms that lie in between floor levels can be represented efficiently (as they represent a prism without any lateral face transitions) we approximately need to represent prisms covering the floor surface area. We assume that we can represent most occurring hexagon prisms with 4 byte values. As a result, the map of a building with a floor surface area of $S_b = 10^4$ m^2 can be stored using $M_b = 40$ KB.

Assuming $S_w = 5 \cdot 10^{10}$ m^2 of indoor area we require roughly $M_w = S_w/S_b \cdot M_b = 2 \cdot 10^{11} =$ 200 gigabytes of memory to store the maps of the indoor world. However, we postulate that we can compress this with loss-less source encoding by at least a factor of four, and much higher compression factors might be achieved if small amounts of distortion can be tolerated.

7.5.3 Communication Effort

We assume that a pedestrian either contributing to the mapping task or with localization needs will carry a device that computes his step vectors (e.g. with an UKF, see Section 2.4) on his mobile device. A server will be in charge of processing these step vectors using FootSLAM. If we can quantize each step vector to 4 bytes, then roughly 4 bytes per second need to be uploaded to the server.

The map merging process (FeetSLAM) may also be performed at the server. Later on, pedestrians can access or download the combined maps of the buildings they visit, with a size of the order of tens of kilobytes.

7.5.4 Privacy Issues

Technologies that help determine the precise position of people allow for an immense number of location-based services. Nevertheless, these technologies also open the door to mischievous usage of these data. Having accurate information about the places that an individual visits and the times when these visits took place can reveal aspects of his private life, for example disabilities, likes and dislikes, place of residence, etc. and could

be used by third parties for unwanted advertising purposes among others. Such privacy issues are not within the scope of this thesis, but we suggest that the same regulations that apply for other location sensors (e.g. GPS) may be applied to the inertial sensor data (see Section IV-A.2 of [128]).

In the personal domain we believe using FootSLAM as a base for localization is clearly an acceptable compromise for indoor navigation since the data collected by inertial sensors do not reveal sensitive information of the environment - as opposed to visual SLAM approaches - such as other people's location.

7.6 Applications and Deployment Scenarios

So far this thesis has addressed many theoretical, implementation and empirical aspects of FootSLAM and FeetSLAM to build a database of probabilistic motion maps. This section explores the main applications of these technologies. We can differentiate between the following two scenarios:

a) Rescue and emergency teams in need of high situation awareness. Knowledge of the map of the environment and the relative position of the team members helps the coordination of the mission at hand, increasing the chances of its success. This usually requires **online** cooperation between the pedestrians.

b) Travelers at an airport or pedestrians in a shopping center that are in need of location-based services. The generation of the map in this case can be done in an **offline** fashion and then served to the users in a later stage.

In the following sections, we will illustrate a number of online and offline applications and present two approaches to implement FeetSLAM: a centralized and a decentralized approach. In addition, we will also explore the use of other sensors and the cooperation with robots to increment robustness, performance and safety.

7.6.1 FeetSLAM Online and Offline Applications

Online Applications

In online applications, a building is mapped by multiple collaborating pedestrians with the objective of providing immediate map and position information to one another or to external users. We envision the following online applications, particularly targeted to security and emergency teams:

- Rescue team coordination: A rescue team enters a building through the same or different entrances. Keeping track of the positions helps obtain a status about already visited areas and helps to find areas which have not been explored yet.

- Firefighters safety: Fire fighters operating under respiratory protection in a smoke-filled building are under extreme stress and high risk of hazardous situations. Knowing their exact locations and the shortest route to get there increases the safety during the operations and would prevent incidents such as the one described in [5]. Figure 7.2 illustrates such a situation fire fighters often need to face.

Figure 7.2: A team of firefighters undertakes a rescue operation under respiratory protection. The generation of a map of the environment they move in and knowing the position of the rest of the team members can be crucial for the success of the operation. Source: US Navy (Licence CC BY 2.0).

- Law enforcement operations: Precise determination of the position of all agents involved in the mission and the knowledge of this position on a map can improve mutual situation awareness and potentially reduce the risk of accidentally harming a team member as well as help coordinate the team.
- Search for missing people in outdoor areas: In areas like deep forest, tunnels or canyons, the use of GNSS is prevented. FeetSLAM can be used to generate a map of the areas as they are visited by the search team and help speed up the search process, specially under bad weather conditions.

In these applications the real-time requirements are severe and usually no prior map data are available, or the map has significantly changed due to the event itself. Doors can be locked or walls might have been destroyed, opening new routes for movement.

Offline Applications

In offline applications we wish to derive a map that can be stored on a server or distributed to localization devices that use it to perform map-aided pedestrian dead reckoning. As more data are collected, the new walks can be incorporated and the maps be refined. Some examples of these offline applications are:

- Mass-market applications: Providing users with localization capabilities and context awareness within museums, public buildings, airports, libraries, etc. FootSLAM maps reflect the frequency of visits of the different areas of a building. This information could help identify possible locations for information stands or advertisements.
- Localization services for visually impaired people: FeetSLAM maps reflect the layout of the walls, the furniture and other obstacles and can be used by people with reading or vision disabilities, possibly through speech translation.
- Support to mobility-impaired people: These individuals might greatly benefit from FootSLAM maps in which the location of ramps and elevators can be indicated. Also, these maps can help find the shortest path to the destination, avoiding long detours that may cause more discomfort to the pedestrian with the walking disability.
- Intelligence services: Intelligence services can use FootSLAM to generate a map of buildings they are interested in. In this kind of applications, not depending on an external preinstalled infrastructure is a valuable characteristic.
- Evacuation routes: Evacuation routes for events with multitudinous attendance can be planed using FeetSLAM. In case of emergency, these routes can be sent to all the mobile terminals, avoiding locked doors and other obstacles. These applications are especially important in public buildings, airports, and any other buildings with high density of people. The maps can help make the crowd move in a coordinate manner and without colliding with each other, following a swarm behavior.
- Rapid response to accidents: A person among a multitudinous group of persons and located in a huge and complex building structure such as a museum, a football stadium or a concert hall, can be reached more quickly thanks to the knowledge of the map of the area, provided the person's location within this map is known. If such person had just suffered a critical incident (e.g. a heart attack), the time to locate him becomes crucial. Furthermore, determining the position of the closest defibrillator, which could also be indicated in the map, can increase the probabilities of saving the person [36].

The next section reviews two alternative approaches to collaboratively build and use crowdsourced maps.

7.6.2 Centralized and Decentralized Approaches

In the centralized approach, the pedestrians only exchange information with a server, but not with each other. In contrast, in the decentralized approach, the pedestrians exchange data directly with each other and compute their own individual map and their own version of the combined map.

Centralized Approach

The pedestrians collaborate to build a crowdsourced map but do not interact with each other directly. The pedestrians can be involved in the combined map generation process in more or less degree:

a) The pedestrians are responsible for collecting the odometry data. In this case, the pedestrians are only equipped with a sensor array.

b) The pedestrians can also generate their own FootSLAM map. To this end, the pedestrians must also carry a device able to perform FootSLAM on the collected odometry data (e.g. a tablet or smartphone).

c) The pedestrians can also generate the combined map using the information that they obtain from the server. In this scenario, the pedestrians need to be equipped with a device able to perform FeetSLAM given all the pedestrian's collected datasets.

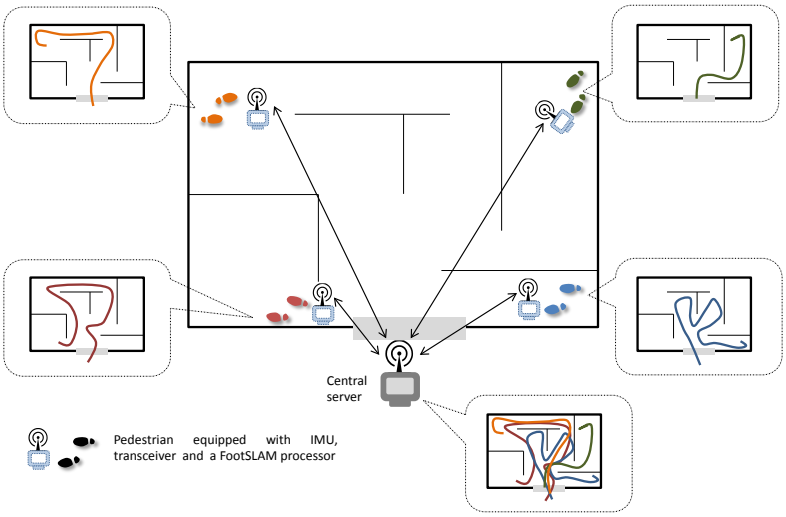


Figure 7.3: Centralized FeetSLAM: Four pedestrians have entered the building through the main entrance. Each one of them is equipped with an IMU, a transceiver and in this case, a FootSLAM processor. All of them are able to generate a map of the areas they have visited, shown by their side. In this case, the central server is in charge of the generation and distribution of the combined map.

Likewise, the server can take on one of the following possibilities:

a) The server takes all individual datasets or FootSLAM maps and is responsible for the generation of the combined map, which can be then distributed to the pedestrian's terminals.

b) The server coordinates the information exchange, but the combined map is computed at each one of the pedestrian's terminal.

The more responsibilities the pedestrians have, the more robust the system is against a failure in the central server or in the communication links. However, this comes at the cost of higher power, memory and computation requirements on the pedestrian's device (e.g. a smartphone).

Figure 7.3 depicts how this centralized approach would work: Each pedestrian has direct access to her own individual map and they communicate with the central server using the available network to compute a combined map with the other pedestrian's maps. The arrows indicate the possibility of communication between a pedestrian and the central server.

Decentralized approach

In this approach, the pedestrians exchange data directly with each other and compute their own individual map and their own version of the combined map. It is possible to exchange odometry data, individual FootSLAM maps or even the combined map.

Figure 7.4 illustrates this decentralized approach for online applications: Each pedestrian has direct access to her own individual map, but to be able to compute a combined map they need to communicate with the other pedestrians using the available network. The arrows indicate the possibility of communication between two pedestrians.

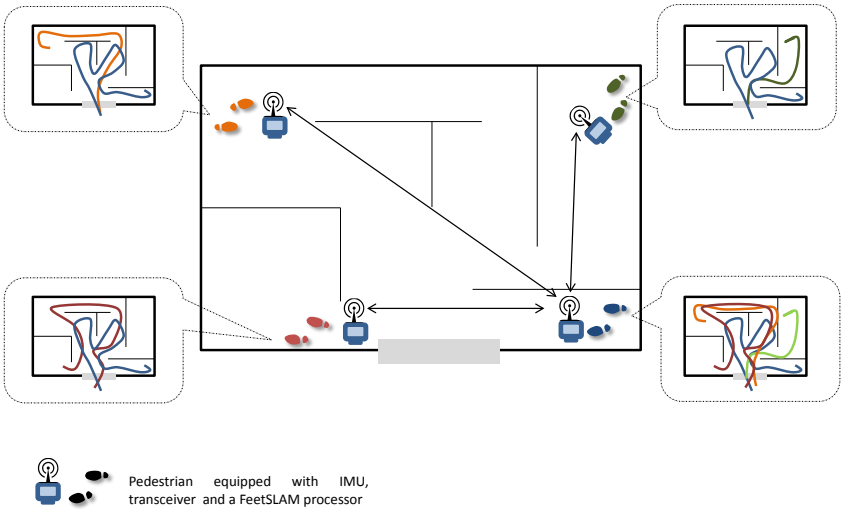


Figure 7.4: Decentralized FeetSLAM: The four pedestrians have now access to the maps of some of the other pedestrians and are able to compute their own partial combined map. This map can be sent to the other pedestrians, helping all pedestrians have knowledge of the position of all the other pedestrians.

Communication Network

The communication can be set up using a centralized network or a decentralized ad hoc network. If the FeetSLAM processing is done centrally, a connection to the central server has to be constantly available, either given implicitly through the centralized network, or through a multihop connection using the decentralized network. If both communication and processing is organized centrally, the system can still benefit from the decentralized network.

A synchronization system needs to be defined and the amount and order of data transmitted (e.g. the whole map obtained by each pedestrian, or only the last computed portion since the last transmission) has to be determined. This is especially important as the pedestrians are constantly moving and the time available for data transmission is limited to the time the two pedestrians are within communication range.

Map information which has most relevance in the direct vicinity of a person can be exchanged with communication partners located in the vicinity. Operational teams are expected to have their own communication network which could be used to distribute maps.

7.6.3 Sensor Fusion Approaches

FootSLAM draws on the exclusive use of inertial sensors, but its framework allows the integration of other sensors, what we call sensor fusion. These new sensors may help anchor the maps to an absolute coordinate system and may help estimate the relative position of the collaborating pedestrians. This can reduce the range of the transformation space to explore when finding the transformation that places all maps within the same coordinate system, reducing drastically the time to combine maps. These are some of the possible techniques:

- Use a GNSS receiver to anchor the map to a global coordinate system when possible (e.g. before entering the building) [131].
- If a WiFi network is available, use WiFi RSS [26] to improve the accuracy of each individual map and anchor the maps to an absolute coordinate system.
- Use the concept introduced by PlaceSLAM [129], whereby recognizable markers are used to help the FootSLAM algorithm converge. These could be markers already available in the environment, like optical features, or these could be markers that are manually placed in or around the building, e.g RFID tags, BLE beacons or UWB stations, but whose position is known. Furthermore, if the position of another pedestrian or a potentially involved robot is known, this information can be used as a form of dynamic PlaceSLAM.
- Estimate the relative position between the pedestrians, which can be inferred when they are in close proximity, for example using Bluetooth scans [93], opportunistic

acoustic ranging [102] or Ultra Wide Band (UWB) ranging [150][112].

- When mapping multistory environments, use an altimeter to estimate the floor at which the pedestrians are located [48].

7.6.4 Human-Robot Collaborative Mapping

There are scenarios in which human intervention is partially prevented, such as a building in danger of collapsing, uncontrolled fire, hazardous materials, etc. In these scenarios, it can be useful to call robots into play [92][48]. The robots can perform some parts of the exploration and undertake the mapping role. The following scenario is envisioned: The pedestrian performing the mapping task enters the building carrying a robot up until a point where the robot is set free. Then, the pedestrian exchanges his current location with the robot, who can start then performing his SLAM process, using visual sensors, PlaceSLAM or any available mapping and localization process. The robot can access and explore those areas where human life is threatened or where human access is not possible. Again, the whole process can be optionally anchored to a global coordinate system using a GNSS receiver before entering the building. Figure 7.5 illustrates this idea.

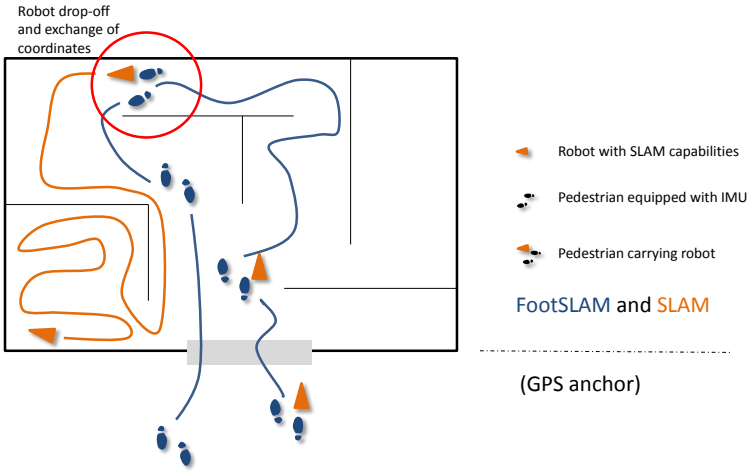


Figure 7.5: Example of human-robot collaborative mapping: A pedestrian carrying a robot enters the building from the outside world optionally anchoring her position to the global coordinate system using a GNSS receiver. Once inside the building, the pedestrian uses FootSLAM to generate a map of the areas she visits and locate herself within it. She reaches a certain point where she drops the robot off, which starts performing SLAM using the last coordinates of the pedestrian as starting point. The pedestrian leaves the building and the robot continues mapping the building autonomously.

7.6.5 Influence on the Collaborating Pedestrians

There exist a number of factors that have an influence on the mapping accuracy of FootSLAM, therefore affecting the resulting combined maps and the speed of the combination process:

- Number of users: the more users, the greater the area that can be potentially mapped, but also the longer it takes to generate a combined map.
- Overlap of the areas visited by the pedestrians: overlap between the individual maps helps to find the best transformation between maps and also increases the convergence when using one as prior of the other. In addition, as shown in Section 3.4, the greater the number of observed spatial transitions at each hexagon's local map, the lower the uncertainty of the map.
- Duration of the walks (extension of the visited area and revisited area): the longer a walk lasts, the more the area is revisited (helping convergence) or the more area is explored (helping extensive mapping).
- Loop closure: closing loops helps the FootSLAM algorithm converge.
- Use of other sensors to anchor the map to an absolute coordinate system and estimate the relative position between the pedestrians, e.g. GNSS [131] or WiFi [26] when available.

Thus, the collaborating pedestrians could be instructed or rewarded to walk for a longer time in certain areas, to close loops, to revisit areas, to flag some markers, to use other sensors, etc. so that a more extensive and accurate combined map is generated [82].

7.7 Discussion

In this chapter we have estimated the time-to-map a given area as a function of the proportion of contributing pedestrians, the frequency of visits of that area and a required number of visits (related to the required accuracy of the map). We have shown that areas that are frequently visited could be mapped in less than a day and areas that are hardly ever visited would take years to be mapped. Nevertheless, we believe that those areas that are frequently visited are the areas that represent possible targets for location-based applications, e.g. airports, underground stations, shopping centers, etc.

Using FeetSLAM, we have estimated that the maps of the entire indoor world could be stored using roughly 200 gigabytes and that the time to compute these maps using tens of walks per area would take less than a month on 5000 cores. These initial estimates suggest the feasibility of our proposed mapping technique to build an indoor map database within an affordable period of time and with relatively sparse user penetration.

In this chapter we have also discussed a number of applications of FootSLAM maps, differentiating between online and offline applications and a centralized and decentralized

approaches with the possibility of using extra sensors or robots.

First responders and mass market applications would highly benefit from deploying these techniques. Future work should address bridging the gap towards a real mapping and localization system and its maintenance and reliability. To do this, crowdsourced step measurement data should be collected by many pedestrians walking normally either in an active or passive manner and processed following a centralized or decentralized approach to validate our proposed approaches.

8 Conclusions and Further Work

8.1 Conclusions

FootSLAM is a technique that estimates probabilistic maps of human motion in constrained environments using only step measurements (a.k.a. human odometry) collected by a walking pedestrian. FeetSLAM extends this idea to multiple pedestrians who collaborate in the mapping task in a crowdsourced manner.

In this thesis, we have extended FootSLAM and FeetSLAM to address the efficient mapping of multistory buildings using only human odometry, in our case obtained with a foot-mounted IMU. We have presented a Bayesian formulation and particle filter implementation of multistory FootSLAM and have applied the algorithm to real-world data. Towards online cooperative FeetSLAM in such environments, we have addressed the reduction in complexity of both FootSLAM and FeetSLAM. To the knowledge of the author, this is the first time to have shown the possibility of mapping large multistory environments in a collaborative fashion using only inertial sensors.

We believe that FootSLAM is a suitable ingredient for mass-market applications where data collected by traveling pedestrians is used to create navigable maps in heavily frequented areas such as airports, stations, shopping centers and other public buildings. Additionally, the FootSLAM map of a building could help optimize the building's operations, usage and infrastructure, by using the frequency of visits to identify hot-spots and bottlenecks, plan alternative routes, extract paths for people with special needs, etc.

Alternatively, the work in this thesis opens the door to using FeetSLAM in an online fashion to increase situation awareness during rescue operations and team operations.

8.1.1 Summary of Contributions

The main contributions of this thesis are the following:

1. Improvement of the mathematical foundation, theoretical background and interpretation of FootSLAM:

    a) Derivation of entropy-based metrics to measure the uncertainty of FootSLAM maps, the uncertainty of the pedestrian's steps and the similarities between two maps or pose sequences.

b) Interpretation of a sequence of pedestrian poses in an environment as a random walk on a weighted graph.

c) Collection of multiple additional human odometry datasets, both in single-story and multistory environments, followed by a validation assessment of FootSLAM.

2. Improvement of the FootSLAM and FeetSLAM implementations towards collaborative real-time mapping of multistory environments:

a) Extension of FootSLAM and FeetSLAM to multistory environments by extending the grid of hexagons to a grid of hexagonal prisms, addressing particle depletion problems such as floor misalignment, deriving a model for the drifting odometry error along the z component, providing an algorithm to compute the floor separation in a building from odometry data, and extending the projection and geometric transformation of FootSLAM maps to also account for the upper and lower prism faces.

b) Implementation of a new map data structure, to reduce the complexity of FootSLAM from $\mathcal{O}(t^2)$ to $\mathcal{O}(t \log t)$, which becomes linear in time for a fixed number of hexagonal bases in the grid and allows real-time mapping of larger areas.

c) Derivation of size-adaptive FootSLAM to map large areas with a grid of hexagonal prisms that grows with the mapping needs of the walk.

d) Reduction of the computational complexity of combining a set of FootSLAM maps by:

i Using an entropy-based metric to select the order in which a set of maps is combined, reducing the dependency with the number of datasets from quadratic to linear.

ii Reducing the complexity of finding the geometric transformation that places two maps within the same coordinate system. By means of the Hough spectrum and spatial spectra of the FootSLAM maps involved, the complexity of the transformation search has been reduced from quadratic to linear in the area of a floor.

3. Analysis of large-scale mapping requirements and indoor pedestrian navigation applications:

a) Study of the time needed to map an area as a function of the proportion of the contributing pedestrians, the frequency of visits of that area and a required number of visits. We have also provided a rough estimate of the memory and time requirements of mapping the whole indoor world.

b) Outline of the main applications and deployment alternatives of crowdsourced FootSLAM maps, both in the mass-market and first responder domains.

8.2 Further Work

Further work could target the following research tasks to improve FootSLAM and FeetSLAM:

1. Towards more realistic scenarios:

   a) Explore additional sensor placements: The walking datasets used in this thesis have been collected with a foot-mounted IMU. Further work should explore collecting step measurements with sensors placed or carried at different body positions, with the goal of moving the collection to more ubiquitous devices such as smartphones, tablets or watches. For example, PocketSLAM [86] has shown how to apply FootSLAM to datasets collected with an IMU placed in the pocket of the user.

   b) Process datasets collected following more realistic walk patterns, in which the pedestrians jump, run or their walks are much shorter (not necessarily "closing the loop") and have few overlapping areas with the other walks.

   c) Include elevators and escalators in the multistory FootSLAM estimation process, for example by detecting such events as proposed in [53].

   d) Expand FootSLAM maps with semantic features. Note that currently FootSLAM and FeetSLAM do not lead to maps that can be directly used in a user facing application such as feature-rich venue map. Research is being directed to actively learn such semantic maps if they are not already available [69][2].

2. Research accuracy improvements:

   a) Investigate the impact of a more complex model of human motion, so far assumed a first order Markov process, in the accuracy of FootSLAM.

   b) Explore alternatives to the existing discretization of the space based on hexagonal bases. A possibility is to use a grid of squares, and store for each square a histogram of angular transition counts [130].

   c) Address failure modes:

      - Open areas: To converge, FootSLAM and FeetSLAM's premise is the presence of walls, doors and obstacles that constrain pedestrians to walk following similar patterns in the environment. In open areas, FootSLAM's mapping accuracy will deteriorate just as much as PDR does. In those cases, relying on the local distortions of the magnetic field as shown in [134] can help achieve sub-meter accuracy even in open areas.

      - Symmetries in the environment: Map-based inertial navigation based only on a FootSLAM motion map might fail in those cases in which there are symmetries in the environment, e.g. parallel aisles on a supermarket that look almost identical in terms of spatial transitions. In those cases, the

use of cameras, BLE beacons or magnetic field measurements could help resolve these ambiguities.

d) Use additional sensors: The experiments in this thesis have been undertaken using only inertial sensors and a magnetometer. However, FootSLAM's framework allows using other sensors following a multi-sensor fusion approach, for example GPS [131], altimeter or WiFi RSS [26]. Using a sensor that provides an absolute reference could help improve the performance of FeetSLAM, anchoring the poses of the pedestrian to a global coordinate system. This would play a very important role in robustifying the search of the transformation that places a set of maps within the same coordinate system.

e) Measure the accuracy of map-aided navigation based on a FootSLAM map by collecting the ground-truth trajectory of the pedestrian during a walk.

3. Multistory online FeetSLAM and maintenance:

a) Extend the entropy metrics to multistory maps.

b) Collect additional multistory datasets within a building and process them with FeetSLAM to further validate the combination of multistory maps.

c) Tackle map exchange for online applications: To experimentally validate the applicability of FeetSLAM to the different scenarios presented in this thesis, further research needs to address the online exchange of maps between the mapping agents or between the agents and a central server.

d) Address FeetSLAM maintenance: This work has focused on FeetSLAM bootstrap, that is, the generation of a global map with no previous information. However, another challenging aspect of mapping is updating and dealing with the mapping of time-variant environments — for example, crowded environments or office rooms with varying furniture layout. We could rely on crowdsourced mapping to collect statistically sufficient data and, similarly to traffic maps, we could generate dynamic maps of a given environment that would reflect its state at different times of the day, the week or the year.

A Appendix

A.1 Autocorrelation Function

The autocovariance of a series z_k with N samples describes the linear dependence of a series with a time-shifted version of itself, i.e. as a function of the time elapsed — called lag — between them [23].

For a stationary process, the autocovariance at lag l is defined as:

$$c_{zz}(l) = E[(z_k - \mu_z)(z_{k+l} - \mu_z)], \tag{A.1}$$

where μ_z is the mean of z_k (constant for a stationary process).

An estimate of the autocovariance, called sample autocovariance is:

$$\hat{c}_{zz}(l) = \frac{1}{N}\sum_{k=1}^{N-l}(z_k - \mu_z)(z_{k+l} - \mu_z),\ \forall\, l \in \{0, \ldots, L\}. \tag{A.2}$$

The sample autocorrelation is then defined as:

$$\hat{\rho}_{zz}(l) = \frac{c_{zz}(l)}{c_{zz}(0)}. \tag{A.3}$$

The sequence of the autocorrelation coefficients $\hat{\rho}_{zz}(l)$ as a function of the lags l is called Autocorrelation Function (ACF) [23]. The following apply:

1. The ACF of an autoregressive (AR) process of order p decays exponentially or as a damped sine wave.
2. For a moving average (MA) process of order q, only the first q coefficients $\rho_{zz}(1) \ldots \rho_{zz}(q)$ are non-zero [23]. In that case, the standard error (σ_ρ) of the ACF for the autocorrelation coefficients of order greater than q is approximately:

$$\sigma_\rho \simeq \frac{1}{\sqrt{N}}\{1 + 2(\hat{\rho}_{zz}(1)^2 + \hat{\rho}_{zz}(2)^2 + \ldots + \hat{\rho}_{zz}(q)^2)\}^{1/2}. \tag{A.4}$$

   Usually, it is convenient to plot two lines about zero at $\pm 2\sigma_\rho$ for the hypothesis in mind, which correspond to the 95% confidence in $\hat{\rho}_{zz}(l)$.

A.2 Partial Autocorrelation Function

Partial autocorrelation refers to the autocorrelation between the series z_k and z_{k+l} after removing any existing dependence within $z_{k+1} \dots z_{k+l-1}$. It can be seen as a function of the autocorrelations. The sequence of the partial autocorrelation coefficients $\phi_{zz}(l)$ as a function of the lags l is called Partial Autocorrelation Function (PACF) [23].

We can obtain an estimation of the PACF (known as sample PACF) by fitting successively AR processes of orders 1, 2, 3,... by least squares and keeping the last coefficient $\hat{\phi}_{zz}(1), \hat{\phi}_{zz}(2), \hat{\phi}_{zz}(3), \dots$ at each stage. The following apply:

1. The PACF of an AR of order p is non-zero at the first p coefficients $\phi_{zz}(1), \dots, \phi_{zz}(p)$. In that case, the standard error (σ_ϕ) of the estimated PACF for the estimated partial autocorrelation coefficients of order greater than p is approximately [23]:

$$\sigma_\phi \simeq \frac{1}{\sqrt{N}}. \tag{A.5}$$

   Usually, it is convenient to plot two lines about zero at $\pm 2\sigma_\phi$ for the hypothesis in mind, which correspond to the 95% confidence in $\hat{\phi}_{zz}(l)$.

2. The PACF of a MA process of order q decays exponentially or following the form of a damped sine.

List of Acronyms and Symbols

Acronyms

ACF Autocorrelation Function

AR autoregressive

ARIMA autoregressive integrated moving average

ARMA autoregressive-moving average

BLE Bluetooth Low Energy

DBN Dynamic Bayesian Network

DLR German Aerospace Center

EKF Extended Kalman Filter

EPA Environmental Protection Agency

GCD Greatest Common Divisor

GNSS Global Navigation Satellite System

GPS Global Positioning System

HS Hough Spectrum

HT Hough Transform

HTC Hexagon Transition Counters

ICP Iterative Closest Point

IMU Inertial Measurement Unit

INS Inertial Navigation System

IR Infrared

KF Kalman Filter

LCP Largest Common Point Set

LoS Line-of-sight

MA moving average

MARU Magnetic Angular Rate Update

MEMS Micro Electromechanical Systems

PACF Partial Autocorrelation Function

pdf probability density function

PDR Pedestrian Dead Reckoning

PF Particle Filter

RBPF Rao-Blackwellized Particle Filter

RFID Radio Frequency Identification

RSS Received Signal Strength

SIS Sequential Importance Sampling

SLAM Simultaneous Localization and Mapping

TUM Technical University of Munich

UKF Unscented Kalman Filter

UWB Ultra Wide Band

WLAN Wireless Local Area Network

ZARU Zero Angular Rate Update

ZUPT Zero Velocity Update

Symbols

General

$p(x)$	Probability density function of x.
$p(x, y)$	Joint probability density function of x and y.
$p(x\|y)$	Conditional probability density function of x given y.
$\mathcal{N}(\boldsymbol{x}; \mu, \sigma)$	Normal distribution of random variable $\boldsymbol{x}$ with mean μ and standard deviation σ.
$\mathcal{N}(\boldsymbol{x}; \mu, \Sigma)$	Multivariate normal distribution of random variable $\boldsymbol{x}$ with mean vector μ and covariance matrix Σ.
$E[\cdot]$	Expectation for the variable in the brackets.
$\delta(l)$	Kronecker delta (which is 1 for lag $l = 0$ and 0 otherwise) or Dirac delta (which is ∞ for $l = 0$ and 0 otherwise).
$\mathbb{R}$	Set of real numbers.
$\mathbb{N}$	Set of natural numbers.
$a \propto b$	a is proportional to b.
$B(\cdot)$	Beta function.
$\Gamma(\cdot)$	Gamma function.
$\Psi(\cdot)$	Digamma function.
$\mathcal{D}(\cdot)$	Dirichlet distribution.
$\mathcal{O}(\cdot)$	Big O notation.
$\lceil \cdot \rceil$	Ceiling function that returns the smallest following integer.
$\lfloor \cdot \rfloor$	Floor function that returns the largest previous integer.

Random Variables

$\boldsymbol{X}$	Random variable.
$\boldsymbol{X}_{0:k}$	The history of the random variable from the beginning up to time instant k.
$\boldsymbol{P}_k$	The pose of a pedestrian at time instant k.
$\boldsymbol{U}_k$	Pedestrian step vector at time instant k, representing the pose change from $\boldsymbol{P}_{k-1}$ to $\boldsymbol{P}_k$.

$\boldsymbol{E}_k$	The (correlated) odometry errors at time instant k.
$\boldsymbol{Z}_k^U$	Step measurement vector at time instant k.
$\mathbf{Vis}_k$	The visual cues and signals that the pedestrian sees at time instant k.
$\mathbf{Int}_k$	The intent of the pedestrian at time instant k.
$\boldsymbol{SC}$	Starting conditions regarding the pose of the pedestrian.
$\boldsymbol{M}$	The map of the environment, which is assumed time invariant.

Hexagonal Grid

N_x	Number of columns in a grid of hexagons or hexagonal prisms.
N_y	Number of rows in a grid of hexagons or hexagonal prisms.
N_z	Number of layers in a grid of hexagons or hexagonal prisms.
N_H	Number of hexagonal bases in a grid of hexagons or hexagonal prisms.
$N_{\mathcal{H}}$	Number of hexagonal prisms in a grid of hexagonal prisms.
H_h	Hexagonal base in a grid, with $0 \leq h \leq N_H - 1$.
H	The set of hexagons $\{H_0, H_1, \ldots, H_{N_H-1}\}$.
r	The radius of a hexagon.
L_l	A layer in a grid of hexagonal prisms, with $0 \leq l \leq N_z - 1$.
$\mathcal{H}_i, \{h, l\}$	A hexagonal prism with hexagonal base H_h at layer L_l, indexed by $i = l \cdot N_H + h$
$\mathcal{H}$	The set of hexagonal prisms $\{\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{N_{\mathcal{H}}-1}\}$.
$H_{\mathcal{H}}$	The height of a hexagonal prism.
e	The edge of a hexagon $0 \leq e \leq 5$.
e_{ij}	Index used to refer to the common edge between two hexagons H_i and H_j.
f	Index used for the faces of a hexagonal prism $0 \leq f \leq 7$.
$(\cdot)_h^e$	Variable for the edge e of hexagon H_h.
$(\cdot)_h^T$	Sum of the variable over all edges of hexagon H_h.
$(\cdot)_{h,l}^f$	Variable for the face f of hexagonal prism $\{h, l\}$.
$(\cdot)_{h,l}^T$	Sum of the variable over all faces of hexagonal prism $\{h, l\}$.
$j \mapsto i$	A neighboring H_j that shares a connecting edge with hexagon H_i.

N_H^G	Number of hexagons in the predefined grid G.
N_G	Number of grids.
G_g	A grid composed of N_H^G hexagons, indexed by g.

Particle Filter Implementation

N_p	Number of particles.
N_{eff}	Effective number of particles.
$[m]$	Index of a particle, with $0 \leq m \leq N_p - 1$
$\{\cdot\}_k^{[m]}$	The superindex and subindex apply to all variables within the curly brackets.
α_h^e	Virtual prior counts for edge e of hexagon H_h.
C_h^e	Number of transitions counted for edge e of hexagon H_h.
K_h^f	Number of transitions counted for face f of hexagon H_h across all layers.
β_h^e	Prior counts from another FootSLAM map for edge e of hexagon H_h.
$w^{[m]}$	The weight associated to particle $[m]$.
$q(\cdot)$	Importance density function.
$I_j^{[m]}$	FootSLAM weight update term for particle $[m]$ when crossing j-th face.
$\mathcal{N}_{\text{vis}}^{[m]}$	Set of hexagons visited by particle m.
$\{M\}^{[m]}$	Data structure used to store the map of particle m.
$(\cdot)^{\text{cum}}$	Superindex used to refer to cumulative posterior maps.
$\boldsymbol{M}^P$	Map learned for a dataset labeled with P.
$H_{\boldsymbol{M}^P}$	Set of hexagons in $\boldsymbol{M}^P$.
$\leftrightarrow$	Variable on the left corresponds to variable on the right.
$\nleftrightarrow$	Variable on the left does not correspond to variable on the right.
S_{floor}	Floor separation.
$\hat{S}_{\text{floor}}$	Estimated floor separation.

Entropy

$\mathrm{h}(\cdot)$	Differential entropy of a continuous random variable.
$\mathrm{H}(\cdot)$	Entropy rate of a discrete stochastic process.

c_{ij}	Conductance between two node i and node j in a weighted graph.
$\vec{P}$	Transition matrix of a Markov chain.
p_{ij}	Element in row i and column j in the $\vec{P}$.
$\vec{x}$	Initial distribution of a Markov Chain.
$\vec{\pi}$	Stationary distribution of an irreducible and aperiodic Markov Chain.
$d_{\mathrm{KL}}(p(\boldsymbol{x})\|\|q(\boldsymbol{x}))$	Relative entropy between two probability density functions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$
$\boldsymbol{H}_{0:k}$	Stochastic process representing the pedestrian being located at a different hexagons over time.

ARIMA Model

p	Autoregressive order.
q	Moving average order.
i	Number of integrations.
B	Backward shift operator.
∇	Backward difference operator.
$\phi(B)$	Stationary autorregressive operator of order p.
$\theta(B)$	Invertible moving average operator of order q.
θ_1, θ_2	Moving average coefficients.
$\gamma_0, \gamma_1, \gamma_2$	Autocovariance coefficients.
c_{xx}	Autocovariance of time series x.

H-tree and Alphabet

n_r	Root node representing the full grid.
n_G	Node that represents all the hexagonal prisms in the predefined grid.
H_{tree}	Height of the H-tree.
H_{2D}	Partial height of the H-tree that maps the hexagonal bases.
H_{3D}	Partial height of the H-tree that maps each hexagonal base to a hexagonal prism.
H_G	Partial height of the 2D subtree that maps the grids.

H_H	Partial height of the 2D subtree that maps each grid to its hexagonal bases.
D	Depth in the tree.
s	Degree of the nodes in the 2D subtree.
s_z	Degree of the nodes in the 3D subtree.
N_n	Number of nodes in the tree.
T	Maximum number of mapped transitions by the alphabet.
S	Number of symbols in the alphabet.
A_s	Symbol number s in the alphabet.

Map Transformation and Combination

N_d	Number of datasets.
$\Delta\theta$	Counterclockwise planar rotation about the center of coordinates.
Δx	Spatial translation along the x axis.
Δy	Spatial translation along the y axis.
Δz	Spatial translation along the z axis.
$\mathcal{P}$	Point set corresponding to a FootSLAM map with hexagonal prism set $\mathcal{H}^P$.
$\mathcal{Q}^{\mathcal{T}}$	Transformed point set $\mathcal{Q}$.
$\mathcal{Q}^{\mathcal{R}}$	Rotated point set $\mathcal{Q}$.
$\mathcal{Q}^{\mathcal{S}}$	Shifted version of rotated point set $\mathcal{Q}^{\mathcal{R}}$.
(ρ, θ)	Polar parametrization of a 2D point.
N_θ	Number of bins used to quantize θ.
N_ρ	Number of bins used to quantize ρ.
Δ_θ^q	With of the bins used for the quantization of θ.
Δ_ρ^q	With of the bins used for the quantization of ρ.
N_r	Number of rotation values applied in the brute-force transformation search.
N_x^{tr}	Number of translation values along the x axis applied in the brute-force transformation search.
N_y^{tr}	Number of translation values along the y axis applied in the brute-force transformation search.
$N_{\max}$	Number of hypotheses kept for rotation and translation values.

List of Figures

List of Tables

List of Author's Publications

Kaiser, S., M. Garcia Puyol and P. Robertson: *Maps-based angular PDFs used as prior maps for FootSLAM.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 113–119, April 2012

Garcia Puyol, M., M. Frassl and P. Robertson: *Collaborative Mapping for Pedestrian Navigation in Security Applications.* In *Future Security*, Bonn, Germany, September 2012

Garcia Puyol, M., P. Robertson and O. Heirich: *Complexity-reduced FootSLAM for indoor pedestrian navigation.* In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–10, Sydney, Australia, November 2012

Garcia Puyol, M., P. Robertson and M. Angermann: *Managing large-scale mapping and localization for pedestrians using inertial sensors.* In *Managing Ubiquitous Communications and Services (MUCS) Workshop of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 121–126, San Diego, California, USA, March 2013

Garcia Puyol, M., P. Robertson and O. Heirich: *Complexity-reduced FootSLAM for indoor pedestrian navigation using a geographic tree-based data structure.* Journal of Location Based Services, 7(3):182–208, September 2013

Robertson, P., M. Frassl, M. Angermann, M. Doniec, B. J. Julian, M. Garcia Puyol, M. Khider, M. Lichtenstern and L. Bruno: *Simultaneous Localization and Mapping for Pedestrians using Distortions of the Local Magnetic Field Intensity in Large Indoor Environments.* In *Indoor Positioning and Indoor Navigation (IPIN)*, Montbeliard, France, October 2013

Khider, M., P. Robertson, M. Frassl, M. Angermann, L. Bruno, M. Garcia Puyol, E. Munoz Diaz and O. Heirich: *Characterization of planar-intensity based heading likelihood functions in magnetically disturbed indoor environments.* In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, Montbeliard, France, October 2013. IEEE

Garcia Puyol, M., D. Bobkov, P. Robertson and T. Jost: *Pedestrian Simultaneous Localization and Mapping in Multistory Buildings Using Inertial Sensors.* IEEE Trans. Intell. Transp. Syst., 15(4):1714–1727, August 2014

Kaiser, S., M. Khider, M. Garcia Puyol, L. Bruno and P. Robertson: *Map-Aided Indoor Navigation.* In *Indoor Wayfinding and Navigation*, chapter 6, page 107.

CRC Press, March 2015

Supervised Master Thesis: Bobkov, D.: *Simultaneous Localisation and Mapping for 3D Pedestrian Navigation based on FootSLAM using Inertial Sensors.* Master's thesis, Technical University of Munich, Germany, 2012

Kaiser, S., M. Garcia Puyol and P. Robertson: *Measuring the Uncertainty of Probabilistic Maps Representing Human Motion for Indoor Navigation.* Mobile Information Systems, August 2016

Garcia Puyol, M., P. Robertson and S. Chakraborty: *Combining Probabilistic Maps of Human Motion Using the Hough Transform.* IEEE Trans. Intell. Transp. Syst., January 2017. Under review

Bibliography

[1] Afzal, M. H., V. Renaudin and G. Lachapelle: *Use of earth's magnetic field for mitigating gyroscope errors regardless of magnetic perturbation.* Sensors, 11(12):11390–11414, November 2011. 2, 24, 25

[2] Alzantot, M. and M. Youssef: *CrowdInside: automatic construction of indoor floorplans.* In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 99–108, November 2012. 5, 7, 157

[3] Alzantot, M. and M. Youssef: *UPTIME: Ubiquitous pedestrian tracking using mobile phones.* In *Wireless Communications and Networking Conference (WCNC),*, pages 3204–3209. IEEE, April 2012. 25

[4] Ambühl, C., S. Chakraborty and B. Gärtner: *Computing Largest Common Point Sets Under Approximate Congruence.* In *Proceedings of the 8th Annual European Symposium on Algorithms*, ESA '00, pages 52–63, London, UK, 2000. Springer-Verlag. 117

[5] Anderson, J. R.: *Abandoned Cold Storage Warehouse Multi-Firefighter Fatality Fire. Technical Report Series.* Technical Report USFA-TR-134/December 1999, U.S. Fire Administration. sHomeland Security, December 1999. 1, 147

[6] Andersone, I.: *The Characteristics of the Map Merging Methods: A Survey.* Scientific Journal of Riga Technical University Computer Science. Applied Computer Systems, 43, January 2010. 5

[7] Android: *Android Wear. Wear what you want.* http://www.android.com/wear/, 2015. 24

[8] Angermann, M. and P. Robertson: *FootSLAM: Pedestrian Simultaneous Localization and Mapping Without Exteroceptive Sensors - Hitchhiking on Human Perception and Cognition.* Proceedings of the IEEE, 100(Special Centennial Issue):1840–1848, May 2012. 6, 7, 27, 30, 36, 37, 73, 88

[9] Anisi, M. and M. Analoui: *Multinomial agent's trust modeling using entropy of the dirichlet distribution.* International Journal of Artificial Intelligence & Applications (IJAIA), 2(3):1–11, July 2011. 58

[10] Apple: *Nike + iPod. Meet your new personal trainer.* http://www.apple.com/ipod/nike/, 2015. 24

[11] Arbel, T. and F. P. Ferrie: *Entropy-based gaze planning.* Image and Vision Computing, 19(11):779–786, September 2001. 49

[12] Bailey, T. and H. Durrant-Whyte: *Simultaneous localization and mapping (SLAM): Part II.* IEEE Robotics & Automation Magazine, 13(3):108–117, September 2006. 19

[13] Bailey, T., J. Nieto and E. Nebot: *Consistency of the FastSLAM algorithm.* In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 424–429, Orlando, Florida, USA, May 2006. 74

[14] Beauregard, S., Widyawan and M. Klepal: *Indoor PDR performance enhancement using minimal map information and particle filters.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, Monterey, California, USA, May 2008. 3

[15] Bellekens, B., V. Spruyt, R. Berkvens and M. Weyn: *A survey of rigid 3D pointcloud registration algorithms.* In *The Fourth International Conference on Ambient Computing, Applications, Services and Technologies*, pages 8–13, 2014. 116

[16] Belongie, S., J. Malik and J. Puzicha: *Shape matching and object recognition using shape contexts.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(4):509–522, April 2002. 117, 138

[17] Berrou, C., A. Glavieux and P. Thitimajshima: *Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1.* In *Communications, IEEE International Conference on*, volume 2, pages 1064–1070. IEEE, May 1993. 41

[18] Besl, P. J. and Neil D. McKay: *A method for registration of 3D shapes.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 14(2):239–256, February 1992. 116

[19] Birk, A. and S. Carpin: *Merging Occupancy Grid Maps From Multiple Robots.* Proceedings of the IEEE, 94(7):1384–1397, July 2006. 5, 127

[20] Bjørke, J. T.: *Framework for entropy-based map evaluation.* Cartography and Geographic Information Systems, 23(2):78–95, January 1996. 49, 50

[21] Blanco, J. L., J. A. Fernandez-Madrigal and J. Gonzalez: *A Novel Measure of Uncertainty for Mobile Robot SLAM with Rao-Blackwellized Particle Filters.* The international Journal of Robotics Research, 27(1):73–89, January 2008. 36, 49, 50

[22] Bobkov, D.: *Simultaneous Localisation and Mapping for 3D Pedestrian Navigation based on FootSLAM using Inertial Sensors.* Master's thesis, Technical University of Munich, Germany, 2012. 30, 88

[23] Box, G. E. P. and G. Jenkins: *Time Series Analysis, Forecasting and Control.* Holden-Day, Incorporated, 1990. 77, 78, 79, 80, 159, 160

[24] Brajdic, A. and R. Harle: *Walk detection and step counting on unconstrained smartphones*. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 225–234, Zurich, Switzerland, September 2013. ACM. 3, 25

[25] Broadband Systems Group Activities YouTube Channel: *Long Foot-SLAM walk outside and around part of our campus*. http://www.youtube.com/watch?v=UxxEZEN1KUE&feature=youtu.be, 2012. 105

[26] Bruno, L. and P. Robertson: *WiSLAM: Improving FootSLAM with WiFi*. In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, Guimaraes, Portugal, September 2011. 3, 5, 7, 129, 151, 153, 158

[27] Carlone, L., M. K. Ng, Jingjing D., B. Bona and M. Indri: *Rao-Blackwellized Particle Filters multi robot SLAM with unknown initial correspondences and limited communication*. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 243–249, May 2010. 5

[28] Carpin, S.: *Fast and accurate map merging for multi-robot systems*. Auton. Robots, 25(3):305–316, October 2008. 5, 115, 117, 119, 122, 134

[29] Castellanos, J. A., J. Neira and J. D. Tardos: *Map building and SLAM algorithms*. Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications, May 2006. 5

[30] Censi, A., L. Iocchi and G. Grisetti: *Scan Matching in the Hough Domain*. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2739–2744, April 2005. 117, 118

[31] Chakraborty, S. and S. Biswas: *Approximation Algorithms for 3D Commom Substructure Identification in Drug and Protein Molecules*. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 253–264, London, UK, 1999. Springer-Verlag. 117

[32] Cinaz, B. and H. Kenn: *HeadSLAM - simultaneous localization and mapping with head-mounted inertial and laser range sensors*. In *Proceedings of IEEE International Symposium on Wearable Computers*, Karlsruhe, Germany, October 2008. 6, 7

[33] Cormen, T. H., C. Stein, R. Rivest and C. Leiserson: *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. 11, 90, 91

[34] Cover, T. M. and J. A. Thomas: *Elements of Information Theory*. John Wiley & Sons, 2006. 48, 50, 51, 53, 54, 58, 59

[35] D. Keralia, K. K. Vyas and K. Deulkar: *Google Project Tango - A Convenient 3D Modeling Device*. International Journal of Current Engineering and Technology, 4(5), October 2014. 6

[36] Dao, T. H. D., Y. Zhou, J. Thill and E. Delmelle: *Spatio-temporal location modeling in a 3D indoor environment: the case of AEDs as emergency medical*

devices. International Journal of Geographical Information Science, 26(3):469–494, March 2012. 148

[37] David A. Levin, Yuval Peres, Elizabeth L. Wilmer: *Markov Chains and Mixing Times*. American Mathematical Society, 1st Edition, 2008. 53, 54

[38] Davison, A. J., I. D. Reid, N. D. Molton and O. Stasse: *MonoSLAM: Real-time single camera SLAM*. IEEE Trans. Pattern Anal. Mach. Intell., 29(6):1052–1067, June 2007. 4

[39] Dinnissen, P., S. N. Givigi and H. M. Schwartz: *Map merging of Multi-Robot SLAM using Reinforcement Learning*. In *Systems, Man, and Cybernetics (SMC), IEEE International Conference on*, pages 53–60, October 2012. 5, 122

[40] Dissanayake, M., P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba: *A solution to the simultaneous localization and map building (SLAM) problem*. Robotics and Automation, IEEE Transactions on, 17(3):229–241, June 2001. 4

[41] Doucet, A., S. Godsill and C. Andrieu: *On sequential Monte Carlo sampling methods for Bayesian filtering*. Statistics and computing, 10(3):197–208, July 2000. 15, 16, 17

[42] Doucet, A.d, N. De Freitas, K. Murphy and S. Russell: *Rao-Blackwellised particle filtering for dynamic Bayesian networks*. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., June 2000. 4, 18, 19

[43] Doyle, P. G. and J. L. Snell: *Random walks and electric networks*. Mathematical Association of America, 1984. 50

[44] Duda, R. O. and P. E. Hart: *Use of the Hough transformation to detect lines and curves in pictures*. Commun. ACM, 15(1):11–15, January 1972. 9, 117, 118, 119

[45] Durrant-Whyte, H. and T. Bailey: *Simultaneous Localization and Mapping: Part I Tutorial*. IEEE Robot. Autom. Mag., 2, June 2006. 4, 18, 19

[46] Estrada, Carlos, Jose Neira and Juan D. Tardos: *Hierarchical SLAM: Real-time accurate mapping of large environments*. Robotics, IEEE Transactions on, 21(4):588–596, August 2005. 111

[47] European Commission: *Indoor air pollution: new EU research reveals higher risks than previously thought*, September 2003. 1

[48] Fallon, M. F., H. Johannsson, J. Brookshire, S. Teller and J. J. Leonard: *Sensor fusion for flexible human-portable building-scale mapping*. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4405–4412, Vilamoura, Algarve, Portugal, October 2012. 6, 7, 26, 71, 82, 152

[49] Faragher, R. and R. Harle: *An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications*. In *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS)*, pages 201–210, Tampa, Florida, September 2014. 2

[50] Ferris, B., D. Fox and N. Lawrence: *WiFi-SLAM Using Gaussian Process Latent Variable Models*. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2480–2485, Hyderabad, India, January 2007. 5, 7

[51] Fischer, C. and H. Gellersen: *Location and Navigation Support for Emergency Responders: A Survey*. Pervasive Computing, IEEE, 9(1):38–47, 2010. 2

[52] Foxlin, E.: *Pedestrian tracking with shoe-mounted inertial sensors*. IEEE Comput. Graph. Appl., 25(6):38–46, November 2005. 3, 24

[53] Frank, K., E. Munoz Diaz, P. Robertson and F. J. Fuentes Sanchez: *Bayesian recognition of safety relevant motion activities with inertial sensors and barometer*. In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 174–184. IEEE, May 2014. 25, 157

[54] Garcia Puyol, M.: *Merging of maps obtained with human odometry based on FootSLAM for pedestrian navigation*. Master's thesis, University of Malaga, Spain, September 2011. 29, 39, 40, 41, 43, 44, 45, 168

[55] Garcia Puyol, M., D. Bobkov and P. Robertson: *Multistory FootSLAM Examples*. http://www.dlr.de/kn/desktopdefault.aspx/tabid-8736/15037_read-37293/, October 2013. 85

[56] Garcia Puyol, M., D. Bobkov, P. Robertson and T. Jost: *Pedestrian Simultaneous Localization and Mapping in Multistory Buildings Using Inertial Sensors*. IEEE Trans. Intell. Transp. Syst., 15(4):1714–1727, August 2014. 9, 69, 89

[57] Garcia Puyol, M., M. Frassl and P. Robertson: *Collaborative Mapping for Pedestrian Navigation in Security Applications*. In *Future Security*, Bonn, Germany, September 2012. 9, 74, 91, 139

[58] Garcia Puyol, M., P. Robertson and M. Angermann: *Managing large-scale mapping and localization for pedestrians using inertial sensors*. In *Managing Ubiquitous Communications and Services (MUCS) Workshop of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 121–126, San Diego, California, USA, March 2013. 4, 9, 139

[59] Garcia Puyol, M., P. Robertson and S. Chakraborty: *Combining Probabilistic Maps of Human Motion Using the Hough Transform*. IEEE Trans. Intell. Transp. Syst., January 2017. Under review. 9, 114

[60] Garcia Puyol, M., P. Robertson and O. Heirich: *Complexity-reduced FootSLAM for indoor pedestrian navigation*. In *Indoor Positioning and Indoor Navi-*

gation (IPIN), 2012 International Conference on, pages 1–10, Sydney, Australia, November 2012. 9, 89, 108

[61] Garcia Puyol, M., P. Robertson and O. Heirich: *Complexity-reduced FootSLAM for indoor pedestrian navigation using a geographic tree-based data structure.* Journal of Location Based Services, 7(3):182–208, September 2013. 9, 74, 89

[62] Georgy, J., A. Noureldin, M.J. Korenberg and M.M. Bayoumi: *Modeling the Stochastic Drift of a MEMS-Based Gyroscope in Gyro/Odometer/GPS Integrated Navigation.* IEEE Trans. Intell. Transp. Syst., 11(4):856–872, December 2010. 77

[63] Godha, S. and G. Lachapelle: *Foot mounted inertial system for pedestrian navigation.* Measurement Science and Technology, 19(7):075202, July 2008. 24

[64] Google Developers: *Project Tango.* https://developers.google.com/project-tango/. 6, 7

[65] Google Inc.: *Indoor Maps – About – Google Maps.* https://www.google.com/maps/about/partners/indoormaps/, July 2014. 3

[66] Google Lat Long: *Making of Maps: The cornerstones.* http://google-latlong.blogspot.com/2014/09/making-of-maps-cornerstones.html, September 2014. 6, 7

[67] Grzonka, S., A. Karwath, F. Dijoux and W. Burgard: *Activity-Based Estimation of Human Trajectories.* IEEE Trans. Robot., 28(1):234–245, February 2012. 6, 7

[68] Guttman, A.: *R-trees: A Dynamic Index Structure for Spatial Searching.* In *International Conference of Management of Data*, pages 47–57, Boston, Massachusetts, USA, 1984. ACM. 92

[69] Hardegger, M., D. Roggen, S. Mazilu and G. Troster: *ActionSLAM: Using location-related actions as landmarks in pedestrian SLAM.* In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, Sydney, Australia, November 2012. 6, 7, 157

[70] Hardegger, M., D. Roggen and G. Troester: *3D ActionSLAM: wearable person tracking in multi-floor environments.* Personal and Ubiquitous Computing, 19(1):123–141, January 2015. 6, 71

[71] Harle, R.: *A Survey of Indoor Inertial Positioning Systems for Pedestrians.* IEEE Commun. Surveys Tuts., 15(3):1281–1293, January 2013. 2, 24

[72] Heckerman, D.: *A tutorial on learning with Bayesian networks.* Technical report, Microsoft Research, March 1995. 33, 34

[73] Hough, P. V. C.: *A method and means for recognizing complex patterns*, December 1962. 117

[74] Howard, A.: *Multi-robot Simultaneous Localization and Mapping using Particle Filters.* International Journal of Robotics Research, 25(12):1243–1256, December 2006. 5, 40

[75] Howe, J.: *The Rise of Crowdsourcing.* Wired, 14(6), June 2006. 5, 39

[76] Illingworth, J. and J. Kittler: *A survey of the Hough transform.* Comput. Vision Graph. Image Process., 44(1):87–116, August 1988. 117

[77] Iocchi, L., S. Pellegrini and G. D. Tipaldi: *Building multi-level planar maps integrating LRF, stereo vision and IMU sensors.* In *IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, pages 1–6, Rome, Italy, September 2007. 70, 71

[78] Jahn, J., U. Batzer, J. Seitz, L. Patino-Studencka and J. Gutierrez Boronat: *Comparison and evaluation of acceleration based step length estimators for handheld devices.* In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6. IEEE, September 2010. 24, 25

[79] Jimenez, A. R., F. Seco, C. Prieto and J. Guevara: *A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU.* In *Intelligent Signal Processing, IEEE International Symposium on*, pages 37–42. IEEE, August 2009. 24

[80] Jimenez, A. R., F. Seco, J. C. Prieto and J. Guevara: *Indoor pedestrian navigation using an INS/EKF framework for yaw drift reduction and a foot-mounted IMU.* In *Positioning Navigation and Communication (WPNC), Workshop on*, pages 135–143, March 2010. 24

[81] Julier, S. J. and J. K. Uhlmann: *A new extension of the Kalman filter to nonlinear systems.* In *Int. symp. aerospace/defense sensing, simul. and controls*, pages 182–193. Orlando, FL, July 1997. 15

[82] Kadous, M. W. and A. Lookingbill: *Quality control of mapping data*, August 6 2013. US Patent 8,504,288. 153

[83] Kaiser, S., M. Garcia Puyol and P. Robertson: *Maps-based angular PDFs used as prior maps for FootSLAM.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 113–119, April 2012.

[84] Kaiser, S., M. Garcia Puyol and P. Robertson: *Measuring the Uncertainty of Probabilistic Maps Representing Human Motion for Indoor Navigation.* Mobile Information Systems, August 2016. 9, 47

[85] Kaiser, S., M. Khider, M. Garcia Puyol, L. Bruno and P. Robertson: *Map-Aided Indoor Navigation.* In *Indoor Wayfinding and Navigation*, chapter 6, page 107. CRC Press, March 2015.

[86] Kaiser, S. and E. Munoz Diaz: *PocketSLAM based on the Principle of the FootSLAM Algorithm*. In *International Conference on Localization and GNSS*, Gothenburg, Sweden, June 2004. 6, 7, 10, 26, 157

[87] Karg, M., K. M. Wurm, C. Stachniss, K. Dietmayer and W. Burgard: *Consistent mapping of multistory buildings by introducing global constraints to graph-based SLAM*. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5383–5388, Anchorage, Alaska, USA, May 2010. 70, 76

[88] Karlsson, N., E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian and M. E. Munich: *The vSLAM Algorithm for Robust Localization and Mapping*. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 24–29, Barcelona, Spain, April 2005. 4

[89] Khider, M.: *Multisensor-Based Positioning for Pedestrian Navigation*. PhD thesis, Ulm Univesity, 2013. 3

[90] Khider, M., P. Robertson, M. Frassl, M. Angermann, L. Bruno, M. Garcia Puyol, E. Munoz Diaz and O. Heirich: *Characterization of planar-intensity based heading likelihood functions in magnetically disturbed indoor environments*. In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, Montbeliard, France, October 2013. IEEE.

[91] Kirkpatrick, S., C. D. Gelatt, M. P. Vecchi et al.: *Optimization by simulated annealing*. Science, 220(4598), May 1983. 42

[92] Kleiner, A., C. Dornhege and S. Dali: *Mapping disaster areas jointly: RFID-Coordinated SLAM by Humans and Robots*. In *Safety, Security and Rescue Robotics, IEEE International Workshop on*, pages 1–6, Gaithersburg, Maryland, USA, September 2007. IEEE. 5, 7, 152

[93] Kloch, K., P. Lukowicz and C. Fischer: *Collaborative PDR Localisation with Mobile Phones*. In *Wearable Computers (ISWC), 2011 15th Annual International Symposium on*, pages 37–40, June 2011. 151

[94] Knoblauch, R. L., M. T. Pietrucha and M. Nitzburg: *Field studies of pedestrian walking speed and start-up time*. Transportation Research Record: Journal of the Transportation Research Board, 1538:27–38, January 1996. 142

[95] Konolige, K., D. Fox, B. Limketkai, J. Ko and B. Stewart: *Map merging for distributed robot navigation*. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 212–217 vol.1, October 2003. 5, 138

[96] Kotanen, A., M. Hannikainen, H. Leppakoski and T. D. Hamalainen: *Positioning with IEEE 802.11b wireless LAN*. In *Personal, Indoor and Mobile Radio Communications, 14th IEEE Proceedings on*, volume 3, pages 2218–2222, September 2003. 2

[97] Krach, B. and P. Robertson: *Cascaded estimation architecture for integration of foot-mounted inertial sensors.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, May 2008. 3, 10, 24, 26

[98] Krach, B. and P. Robertson: *Integration of foot-mounted inertial sensors into a Bayesian location estimation framework.* In *WPNC*, pages 55–61. IEEE, March 2008. 3

[99] LaMarca, A., Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter and J. Tabert: *Place lab: Device positioning using radio beacons in the wild.* In *Pervasive computing*, pages 116–133. Springer, January 2005. 2

[100] Leonard, J. J. and H. F. Durrant-Whyte: *Simultaneous map building and localization for an autonomous mobile robot.* In *Intelligent Robots and Systems. 'Intelligence for Mechanical Systems, Proceedings IROS. IEEE/RSJ International Workshop on*, pages 1442–1447 vol.3, November 1991. 4, 20

[101] Li, B., B. Harvey and T. Gallagher: *Using barometers to determine the height for indoor positioning.* In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–7. IEEE, October 2013. 26

[102] Li, Y., G. Chen and M. Sun: *An Indoor Collaborative Pedestrian Dead Reckoning System.* In *Parallel Processing (ICPP), 42nd International Conference on*, pages 923–930, October 2013. 152

[103] Lovász, L.: *Random walks on graphs: a survey.* Combinatorics, Paul Erd"os is Eighty, Keszthely (Hungary), 2(1):1–46, May 1993. 50

[104] Madgwick, Sebastian Oh: *An efficient orientation filter for inertial and inertial/magnetic sensor arrays.* Report x-io and University of Bristol (UK), April 2010. 25

[105] McEliece, Robert J., D. J. C. MacKay and J. Cheng: *Turbo decoding as an instance of Pearl's "belief propagation" algorithm.* Selected Areas in Communications, IEEE Journal on, 16(2):140–152, February 1998. 42

[106] Millonig, A. and K. Schechtner: *Developing Landmark-Based Pedestrian-Navigation Systems.* IEEE Trans. Intell. Transp. Syst., 8(1):43–49, March 2007. 4

[107] Mirowski, P., T. K. Ho, S. Yi and M. MacDonald: *SignalSLAM: Simultaneous Localization and Mapping with Mixed WiFi, Bluetooth, LTE and Magnetic Signals.* In *Indoor Positioning and Indoor Navigation (IPIN)*, Montbeliard, France, October 2013. 5, 7

[108] Montemerlo, Michael, Sebastian Thrun, Daphne Koller and Ben Wegbreit: *FastSLAM: A factored solution to the simultaneous localization and mapping problem.* In *Proc. AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, July 2002. 4, 20, 21, 22, 23, 29, 91, 92, 95, 96

[109] Munoz Diaz, E. and A. L. Mendiguchia Gonzalez: *Step Detector and Step Length Estimator for an Inertial Pocket Navigation System.* In *Indoor Positioning and Indoor Navigation (IPIN)*, Bexco, Busan, Korea, October 2014. 25

[110] Munoz Diaz, E., A. L. Mendiguchia Gonzalez and F. de Ponte Mueller: *Standalone inertial pocket navigation system.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 241–251, Monterey, California, USA, May 2014. 3, 24, 25

[111] Murphy, K. P.: *Machine learning: a probabilistic perspective.* MIT Press, 2012. 17

[112] Nilsson, J., D. Zachariah, I. Skog and P. Haendel: *Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging.* EURASIP Journal on Advances in Signal Processing, April 2013. 152

[113] Nilsson, J.-O., I. Skog, P. Handel and K. Hari: *Foot-mounted INS for everybody - an open-source embedded implementation.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, pages 140–145, Myrtle Beach, South Carolina, USA, April 2012. 3, 24

[114] Open Street: *Open Street Map.* http://www.android.com/wear/, July 2014. 3

[115] Oravetz, A. S. and H. J. Sandberg: *Stationary and nonstationary characteristics of gyro drift rate.* AIAA Journal, 8(10):1766–1772, October 1970. 77

[116] Ozkucur, N. E. and H. L. Akin: *Cooperative Multi-robot Map Merging Using Fast-SLAM.* In Baltes, Jacky, Michail G. Lagoudakis, Tadashi Naruse and Saeed Shiry Ghidary (editors): *RoboCup 2009*, pages 449–460. Springer-Verlag, Berlin, Heidelberg, January 2010. 5, 122

[117] Penny, W. D.: *KL-Divergences of Normal, Gamma, Dirichlet, and Wishart Densities*, March 2001. 60

[118] Powers, E. and A. Colina: *Wide Area Network Synchronization Using Locata.* In *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, pages 3434–3442, Tampa, Florida, September 2015. 2

[119] Priestley, M. B.: *Spectral Analysis and Time Series.* Academic Press, 1981. 81

[120] Qian, J., J. Ma, L. Xu, R. Ying, W. Yu and P. Liu: *Investigating the use of MEMS Based Wrist-worn IMU for Pedestrian Navigation Application.* In *ION GNSS*, Nashville, Tennessee, USA, September 2013. 3, 24

[121] Rached, Z., F. Alajaji and L. L. Campbell: *The Kullback-Leibler divergence rate between Markov sources.* Information Theory, IEEE Transactions on, 50(5):917–921, May 2004. 54, 60

[122] Rai, A., K. K. Chintalapudi, V. N. Padmanabhan and R. Sen: *Zee: Zero-effort crowdsourcing for indoor localization.* In *Proceedings of the 18th annual inter-*

national conference on Mobile computing and networking, pages 293–304, Istanbul, Turkey, August 2012. ACM. 4

[123] Rajagopal, S.: *Personal dead reckoning system with shoe mounted inertial sensors*. Master's thesis, KTH, 2008. 25

[124] Rapf, O.: *Europes's buildings under the microscope*. In *Buildings Performance Institute Europe (BPIE)*. BPIE, 2011. 141

[125] Rehrl, K., S. Bruntsch and H. J. Mentz: *Assisting Multimodal Travelers: Design and Prototypical Implementation of a Personal Travel Companion*. IEEE Trans. Intell. Transp. Syst., 8(1):31–42, March 2007. 4

[126] Renaudin, V., V. Demeule and M. Ortiz: *Adaptative pedestrian displacement estimation with a smartphone*. In *Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–9, October 2013. 25

[127] Renaudin, V., M. Susi and G. Lachapelle: *Step Length Estimation Using Handheld Inertial Sensors*. Sensors, 12(7):8507–8525, June 2012. 24, 25

[128] Robertson, P. and M. Angermann: *Simultaneous Localization and Mapping for Pedestrians using only Foot-Mounted Inertial Sensors*. In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, Orlando, Florida, USA, 2009. 6, 8, 27, 28, 29, 30, 32, 34, 35, 37, 73, 74, 115, 146

[129] Robertson, P., M. Angermann and M. Khider: *Improving Simultaneous Localization and Mapping for Pedestrian Navigation and Automatic Mapping of Buildings by using Online Human-Based Feature Labeling*. In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, Palm Springs, California, USA, May 2010. 3, 151

[130] Robertson, P., M. Angermann and B. Krach: *Method for creating a map relating to location-related data on the probability of future movement of a person*, January 2014. US Patent 8,626,443. 157

[131] Robertson, P., M. Angermann, B. Krach and M. Khider: *Inertial based joint mapping and positioning for pedestrian navigation*. In *ION GNSS*, Savannah, Georgia, USA, September 2009. 3, 30, 45, 129, 151, 153, 158

[132] Robertson, P., M. Frassl, M. Angermann, M. Doniec, B. J. Julian, M. Garcia Puyol, M. Khider, M. Lichtenstern and L. Bruno: *3D MagSLAM walk at DLR*. https://www.youtube.com/watch?v=U9tC1IbPi-c, October 2013. 88, 110

[133] Robertson, P., M. Frassl, M. Angermann, M. Doniec, B. J. Julian, M. Garcia Puyol, M. Khider, M. Lichtenstern and L. Bruno: *Magnetic Localization in Buildings and MagSLAM*. www.kn-s.dlr.de/indoornav/magnetic.html, October 2013. 88

[134] Robertson, P., M. Frassl, M. Angermann, M. Doniec, B. J. Julian, M. Garcia Puyol, M. Khider, M. Lichtenstern and L. Bruno: *Simultaneous Localization and Mapping for Pedestrians using Distortions of the Local Magnetic Field Intensity in Large Indoor Environments*. In *Indoor Positioning and Indoor Navigation (IPIN)*, Montbeliard, France, October 2013. 37, 73, 74, 88, 110, 138, 157

[135] Robertson, P., M. Frassl, M. Angermann, M. Doniec, B. J. Julian, M. Garcia Puyol, Mo. Khider, M. Lichtenstern and L. Bruno: *3D MagSLAM walk at TUM*. https://www.youtube.com/watch?v=cBzOCzyoqSU, October 2013. 88, 110

[136] Robertson, P., M. Garcia Puyol and M. Angermann: *Collaborative Pedestrian Mapping of Buildings Using Inertial Sensors and FootSLAM*. In *ION GNSS*, Portland, Oregon, USA, September 2011. 6, 7, 8, 39, 40, 41, 42, 43, 44, 45, 46, 64, 65, 74, 114, 115, 122, 123, 127, 129, 134, 138, 169

[137] Sanjeev Arulampalam, M., S. Maskell, N. Gordon and T. Clapp: *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking*. IEEE Trans. Signal Process., 50(2):174–188, February 2002. 3, 4, 6, 12, 13, 14, 16, 17, 18, 40

[138] Sayood, Khalid: *Introduction to Data Compression*. M. Kaufmann, Third edition, 2005. 103

[139] Seco-Granados, G., J. A Lopez-Salcedo, D. Jimenez-Banos and G. Lopez-Risueno: *Challenges in Indoor Global Navigation Satellite Systems: Unveiling its core features in signal processing*. Signal Processing Magazine, IEEE, 29(2):108–131, March 2012. 2

[140] Shannon, C. E.: *A mathematical theory of communication*. The Bell System Technical Journal, 27:379–423, 623–656, 1948. 48, 49

[141] Shen, G., Z. Chen, P. Zhang, T. Moscibroda and Y. Zhang: *Walkie-Markie: indoor pathway mapping made easy*. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, pages 85–98, Berkeley, CA, USA, April 2013. USENIX Association. 4, 5, 7

[142] Shin, B., J. H. Lee, H. Lee, E. Kim, J. Kim, S. Lee, Y. Cho, S. Park and T. Lee: *Indoor 3D pedestrian tracking algorithm based on PDR using smartphone*. In *12th International Conference on Control, Automation and Systems (ICCAS)*, pages 1442–1445, ICC Jeju, Korea, October 2012. 71, 76

[143] Shin, H., Y. Chon and H. Cha: *Unsupervised Construction of an Indoor Floor Plan Using a Smartphone*. IEEE Trans. Syst., Man, Cybern. C, 42(6):889–898, November 2012. 5, 7

[144] Smith, G., R. Wieser, J. Goulding and D. Barrack: *A refined limit on the predictability of human mobility*. In *Pervasive Computing and Communications*

(PerCom), 2014 IEEE International Conference on, pages 88–94. IEEE, March 2014. 50

[145] Smith, R., M. Self and P. Cheeseman: *Estimating uncertain spatial relationships in robotics*. Autonomous Robot Vehicles, pages 167–193, 1990. 4, 20

[146] Smith, R. C. and P. Cheeseman: *On the representation and estimation of spatial uncertainty*. The international journal of Robotics Research, 5(4):56–68, December 1986. 4, 20

[147] Song, C., Z. Qu, N. Blumm and A. L. Barabási: *Limits of predictability in human mobility*. Science, 327(5968):1018–1021, February 2010. 50

[148] Stachniss, C., G. Grisetti and W. Burgard: *Information Gain-based Exploration Using Rao-Blackwellized Particle Filters*. In *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, June 2005. 49

[149] Steinhoff, U. and B. Schiele: *Dead reckoning from the pocket - An experimental study*. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 162–170, Mannheim, Germany, March 2010. 3, 24

[150] Strömback, P., J. Rantakokko, S. Wirkander, M. Alexandersson, K. Fors, I. Skog and P. Händel: *Foot-Mounted Inertial Navigation and Cooperative Sensor Fusion for Indoor Positioning*. In *Proceedings of the 2010 international technical meeting of the Institute of Navigation*. INST NAVIGATION, January 2010. 152

[151] Susi, M., V. Renaudin and G. Lachapelle: *Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users*. Sensors, 13(2):1539–1562, January 2013. 3, 25, 88

[152] Thrun, S.: *A Probabilistic Online Mapping Algorithm for Teams of Mobile Robots*. International Journal of Robotics Research, 20, May 2001. 5

[153] Thrun, S.: *Robotic Mapping: A Survey*. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, February 2002. 4

[154] Thrun, S., W. Burgard and D. Fox: *A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping*. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 321–328 vol.1, 2000. 4

[155] Thrun, S., W. Burgard and D. Fox: *Probabilistic robotics*. MIT Press, 2005. 11, 12, 15, 19, 20, 21, 23

[156] Thrun, S., D. Koller, Z. Ghahramani, H. Durrant-Whyte and A. Y. Ng: *Simultaneous mapping and localization with sparse extended information filters: Theory and initial results*. Springer, 2004. 20, 21

[157] Thrun, S. and M. Montemerlo: *The graph SLAM algorithm with applications to large-scale mapping of urban structures.* The International Journal of Robotics Research, 25(5-6):403–429, May 2006. 4, 21, 22

[158] Topal, S., I. Erkmen and A. M. Erkmen: *A Novel Map Merging Methodology for Multi-Robot Systems.* In *Proceedings of the World Congress on Engineering and Computer Science, WCECS*, San Francisco, USA, October 2010. 138

[159] Unite for Sight: *Global Eye Health Statistics.* http://www.uniteforsight.org/eye_stats.php, 2009. 1

[160] U.S. Environmental Protection Agency: *Report to Congress on indoor air quality: Volume 2*, 1989. 1

[161] Vaughan, C. L., B. L. Davis and J. O'Connor: *Dynamics of human gait.* Human Kinetics Publishers, January 1992. 24

[162] Welch, G. and G. Bishop: *An Introduction to the Kalman Filter.* Technical Report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995. 6, 13, 14, 15

[163] Wendlandt, K., K. Sukchaya, P. Robertson, M. Khider and M. Angermann: *Demonstration of a Realtime Active-Tag RFID, Java Based Indoor Localization System using Particle Filtering.* In *Adjunct Proceedings of the Ubicomp*, Innsbruck, Austria, 2007. 2

[164] Woodman, O. and R. Harle: *Pedestrian localization for indoor environments.* In *Proceedings of the International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, Seoul, South Korea, September 2008. 3

[165] Woodman, O. J.: *An introduction to inertial navigation.* Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007. 3, 24, 77

[166] Zampella, F., M. Khider, P. Robertson and A. Jimenez: *Unscented Kalman Filter and Magnetic Angular Rate Update (MARU) for an Improved Pedestrian Dead-Reckoning.* In *IEEE/ION Position Location and Navigation Symposium (PLANS)*, Myrtle Beach, South Carolina, USA, April 2012. 2, 3, 24, 25, 26, 27, 37, 61, 77, 78, 84, 131, 168

[167] Zipf, G.: *Human Behaviour and the Principle of Least-Effort.* Journal of Clinical Psychology, 1949. 142