Pedestrian localisation for indoor environments

A dissertation submitted for the degree
of Doctor of Philosophy

Oliver J. Woodman
St Catharine's College

September 9, 2010

DIGITAL TECHNOLOGY GROUP
Computer Laboratory
University of Cambridge

Typeset in Times with LaTeX

This dissertation is the result of my own work and includes nothing that is the outcome of work done in collaboration except where specifically indicated in the text.

No part of this dissertation has already been, or is currently being submitted by the author for any other degree or diploma or other qualification.

This dissertation does not exceed 60,000 words including tables and footnotes, but excluding appendices, bibliography, photographs and diagrams.

This work was supported by the EPSRC.

Abstract

Pedestrian localisation for indoor environments

Oliver J. Woodman

Ubiquitous computing systems aim to assist us as we go about our daily lives, whilst at the same time fading into the background so that we do not notice their presence. To do this they need to be able to sense their surroundings and infer context about the state of the world. Location has proven to be an important source of contextual information for such systems. If a device can determine its own location then it can infer its surroundings and adapt accordingly.

Of particular interest for many ubiquitous computing systems is the ability to track people in indoor environments. This interest has led to the development of many indoor location systems based on a range of technologies including infra-red light, ultrasound and radio. Unfortunately existing systems that achieve the kind of sub-metre accuracies desired by many location-aware applications require large amounts of infrastructure to be installed into the environment.

This thesis investigates an alternative approach to indoor pedestrian tracking that uses on-body inertial sensors rather than relying on fixed infrastructure. It is demonstrated that general purpose inertial navigation algorithms are unsuitable for pedestrian tracking due to the rapid accumulation of errors in the tracked position. In practice it is necessary to frequently correct such algorithms using additional measurements or constraints. An extended Kalman filter is developed for this purpose and is applied to track pedestrians using foot-mounted inertial sensors. By detecting when the foot is stationary and applying zero velocity corrections a pedestrian's relative movements can be tracked far more accurately than is possible using uncorrected inertial navigation.

Having developed an effective means of calculating a pedestrian's relative movements, a localisation filter is developed that combines relative movement measurements with environmental constraints derived from a map of the environment. By enforcing constraints such as impassable walls and floors the filter is able to narrow down the absolute position of a pedestrian as they move through an indoor environment. Once the user's position has been uniquely determined the same filter is demonstrated to track the user's absolute position to sub-metre

accuracy.

The localisation filter in its simplest form is computationally expensive. Furthermore symmetry exhibited by the environment may delay or prevent the filter from determining the user's position. The final part of this thesis describes the concept of assisted localisation, in which additional measurements are used to solve both of these problems. The use of sparsely deployed WiFi access points is discussed in detail.

The thesis concludes that inertial sensors can be used to track pedestrians in indoor environments. Such an approach is suited to cases in which it is impossible or impractical to install large amounts of fixed infrastructure into the environment in advance.

Acknowledgements

Many people have helped me to complete the research presented in this thesis. In particular I wish to thank Andy Hopper for his insight, guidance and financial assistance. I also wish to thank Robert Harle for giving up a great deal of his time to help with my research. Similarly Andrew Rice, Ripduman Sohan, Alistair Beresford and Brian Jones have all contributed their time and knowledge to assist me during the last three years.

In addition to the individuals named above, the Digital Technology Group as a whole has provided a friendly and vibrant atmosphere in which it was a pleasure to work. Their diverse knowledge has helped to broaden my education and their feedback has helped to shape the direction of my research.

I wish to thank my examiners Peter Robinson and Paul Groves for their comments and suggestions, which have enabled me to make significant improvements to the quality of this manuscript.

Finally I wish to thank my parents, grandparents, family and friends for their love and support.

Publications

Oliver Woodman. An Introduction to Inertial Navigation. Technical Report 696, University of Cambridge, Computer Laboratory, August 2007.

Oliver Woodman and Robert Harle. Pedestrian Localisation for Indoor Environments. In *Proceedings of the 10th International Conference on Ubiquitous Computing (Ubicomp 2008), Seoul, South Korea*, September 2008.

Oliver Woodman and Robert Harle. Rf-based Initialisation for Inertial Pedestrian Tracking. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive 2009), Nara, Japan*, May 2009.

Oliver Woodman and Robert Harle. Concurrent scheduling in the Active Bat location system. In *Proceedings of the Sixth IEEE International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2010), Mannheim, Germany*, March 2010.

Andrew Rice and Oliver Woodman. Crowd-sourcing World Models with OpenRoomMap. In *Proceedings of the Eighth IEEE International Conference on Pervasive Computing and Communications (PerCom 2010), Work in Progress, Mannheim, Germany*, March 2010.

Contents

List of figures

List of tables

Chapter 1

Introduction

As computers become smaller, cheaper and more powerful, they are becoming increasingly common in our everyday environment. Computers can now be found in light switches, door locks, coffee machines, microwaves and fridges. There are multiple computers in most modern cars, controlling everything from a vehicle's traction and braking to its interior lights and sound system. These computers assist us as we go about our lives, often without us noticing their presence. This is known as *ubiquitous computing*, a term first used by Mark Weiser in the late 1980s. In his seminal paper on the subject he states that "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." [1].

For ubiquitous computers to disappear in the way envisaged by Mark Weiser, they must be able to infer some context about the environment into which they are placed. For example a computer controlled light switch is no better than a physical one if it is unable to sense whether there are people in the room or whether it is light or dark. In contrast a light switch that is able to sense these properties can automatically switch on when a person enters a darkened room and off once the room has been empty for a certain period of time.

There are many types of contextual information that can be used by ubiquitous computers. Location information is one of the most important, particularly now that mobile computing platforms are cheap and readily available. A computer that is able to determine its location can adapt to its local environment. For example if a mobile telephone senses that it is in a lecture theatre then it may choose to automatically switch to silent mode, avoiding a disturbance should an incoming call be received. Location information alone is not sufficient for this task. It is also necessary to have access to a *world model* that can be used to give meaning to a location [2]. In this example a suitable world model consists of a map of the building annotated with spatial zones that indicate regions within which audible disturbances should be suppressed.

The branch of ubiquitous computing in which the primary source of context is location (together with a world model) is often referred to as *location-aware computing* [3]. Perhaps the most widespread example of location-aware computing today is satellite navigation, in which location information obtained using a global navigation satellite system (GNSS) is used together with a map of the road network in order to provide turn-by-turn navigation for a vehicle [4].

Whilst GNSSs allow devices to position themselves when outdoors, they rarely support indoor positioning because the satellite signals used are too weak to penetrate most buildings. As a result indoor positioning systems have been developed based on a variety of other technologies including infra-red [5], ultrasound [6, 7, 8] and radio [9, 10, 11, 12]. Of particular interest for indoor location-aware applications has been the tracking of people, which is also the focus of this thesis. For example the Active Badge location system was used to track employees at Olivetti research in Cambridge [5]. This allowed telephone calls to be automatically forwarded to the rooms within which the intended recipients were located, as well as allowing computer terminals to automatically retrieve the preferences of whoever sat in front of them. Despite the fact that indoor positioning systems enable the development of these and many more location-aware applications, their adoption to date has been extremely limited.

One of the biggest problems preventing the widespread adoption of indoor positioning systems is the correlation that currently exists between the accuracy of such systems and their cost and infrastructure requirements. Nearly all indoor location-aware applications need to know the room within which a person or device is located. For pedestrian tracking metre-level accuracy is often required so that an application can determine which objects in the environment are close enough to the user for a physical interaction to take place (e.g. whether a user is positioned close enough to a computer to be using it). Unfortunately existing indoor positioning systems that are able to achieve such accuracies require large amounts of infrastructure to be installed into the environment. For example some ultrasound-based systems are able to position devices to within a few centimetres [6], but remain prohibitively expensive and rely on large amounts of fixed infrastructure. Such infrastructure is not only labour intensive to install, but is also expensive to maintain. At the other end of the scale are systems based on technologies such as WiFi [10, 11, 12]. Although such systems are far cheaper and require less infrastructure, they are often unable to reliably determine even the room within which a person or device is located in real world deployments. Before indoor positioning systems can be widely adopted it will be necessary to develop positioning technologies that are accurate enough for the majority of location-aware applications whilst at the same time having minimal cost and infrastructure requirements.

1.1 The potential of inertial sensors

In theory it is possible for an object to keep track of changes in its own position using an inertial measurement unit (IMU) containing linear accelerometers and gyroscopes. The basic principle is to track changes in the device's orientation using the gyroscopes, which can be used to project the accelerations measured locally by the device into a global frame of reference. These accelerations can then be double integrated to track changes in the object's position [13]. In practice however, errors rapidly accumulate in the tracked position due to the propagation of measurement errors through the projection and integration calculations. Such errors are collectively referred to as *drift*. Despite the accumulation of drift, inertial navigation systems are commonly used in aeroplanes, missiles and submarines. Aeroplanes and missiles can use GNSSs to periodically correct drift, however this is not possible for a submerged submarine because the satellite signals used do not penetrate far underwater. Hence extremely accurate inertial measurement units are required. Those deployed in modern submarines are accurate enough to limit drift to within a few nautical miles after one day of operation, however such devices are large and heavy, in some cases measuring metres across.

Until recently it was not possible to manufacture IMUs that were small and light enough to be used for pedestrian tracking, however recent advances in the construction of micro-machined electromechanical systems (MEMS) have now made it possible to manufacture small inertial sensor chips from which such devices can be built. MEMS inertial sensors are not as accurate as the large mechanical devices used in submarines, but they are far cheaper and contain very few moving parts, making them extremely durable. The quality of MEMS inertial sensors is improving rapidly and their price continues to fall, partly driven by their mass manufacture for use in many of today's modern mobile telephones (as user input devices) and laptops (to protect hard disks by detecting sudden accelerations). This has allowed the development of relative positioning systems that are able to track the movement of pedestrians relative to their initial positions and headings [14, 15, 16]. Such systems are able to operate both outside and indoors, however without periodic correction their accuracy degrades over time. This thesis investigates the use of environmental constraints as a means of providing such corrections when tracking indoors. In particular the movement of a pedestrian is naturally constrained by walls, floors and ceilings that are present in nearly all indoor environments. By using such constraints it may be possible to track a pedestrian indefinitely in an indoor environment, with no degradation in accuracy and without the need to install any fixed infrastructure. Hence this approach could enable the development of new indoor location systems that are suitable for use when the installation of such infrastructure is either uneconomical, impractical or impossible.

This thesis investigates the development of an indoor pedestrian location system based on the ideas outlined above. In doing so it aims to answer the following questions:

1. **How rapidly does drift accumulate in an inertial navigation system when using current state-of-the-art MEMS inertial sensors?** The aim here is to determine the extent to which it is possible to track the relative movements of an object or person using modern MEMS inertial sensors together with general purpose inertial navigation algorithms.

2. **How accurately can the relative movement of a pedestrian be measured using MEMS inertial sensors?** This question differs to the one above because it deals specifically with the tracking of people. It may be possible to exploit domain specific knowledge in order to do better than the general case.

3. **What is the minimum infrastructure or knowledge required to determine a pedestrian's absolute location and heading?** A relative tracking system based on inertial sensors can only be used to track a user's absolute position if it is provided with their initial position and heading. The aim is to find a way in which this initial state can be determined that does not rely on the installation of large amounts of fixed infrastructure (and preferably none at all).

4. **Can environmental constraints be used to prevent the accumulation of drift over time?** This would allow the position of an individual to be tracked for an indefinite period of time. If this is possible, to what accuracy can the individual's position be tracked?

1.2 Thesis outline

The structure of the remainder of this thesis is outlined below. An indoor pedestrian tracking system is developed throughout in order to answer the questions posed above. The major components of the resulting system are shown in Figure 1.1, which is annotated to indicate the chapters within which each of the components are described.

Chapter 2 outlines related work and motivates the remainder of this thesis. Existing positioning technologies, robotics research and concurrent work on pedestrian localisation are described.

Chapter 3 describes the error characteristics of MEMS inertial sensors, introduces the fundamentals of inertial navigation and outlines the propagation of errors through inertial navigation systems. A general purpose extended Kalman filter is presented that is able to correct such systems using external measurements and constraints.

Chapter 4 describes the application of inertial navigation techniques to the problem of pedestrian dead reckoning, in which the aim is the track a pedestrian's relative movements over

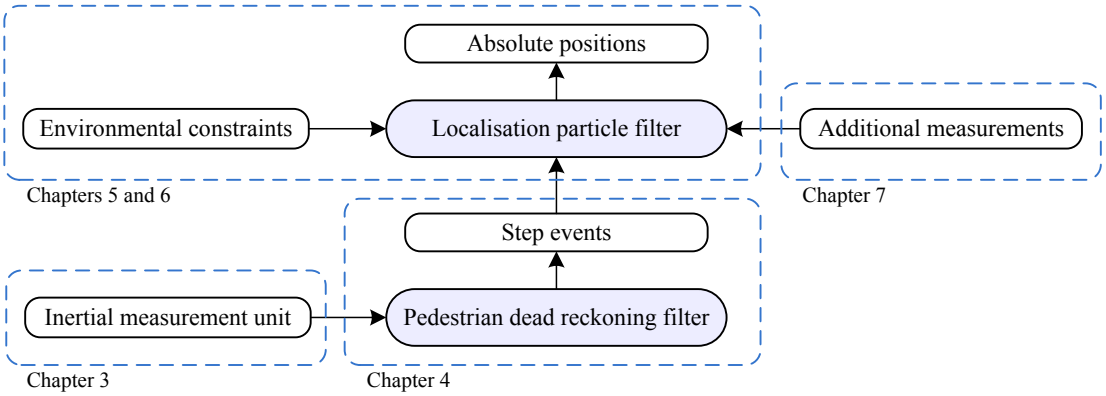


Figure 1.1: The components of the pedestrian tracking system developed throughout this thesis. Annotations indicate the chapter(s) in which each component is described.

time. A pedestrian dead reckoning filter is developed based on the extended Kalman filter presented in Chapter 3.

Chapter 5 investigates how knowledge of environmental constraints can be combined with relative movement information in order to determine and subsequently track the absolute position of a pedestrian. A particle filter implementation is developed, which is able to localise and subsequently track a user by combining relative movement information with constraints defined by a map of the environment.

Chapter 6 describes a number of optimisations that can be made to the algorithms developed in Chapter 5. The performance and error characteristics of the resulting system are investigated in detail.

Chapter 7 introduces the concept of assisted localisation, in which additional measurements and constraints can be used to reduce the cost of localisation. WiFi-assisted localisation is described in detail and is implemented as an extension to the algorithms developed in Chapters 5 and 6.

Chapter 8 revisits the research questions posed earlier in this chapter, outlines possible avenues for future research and summarises the main contributions of this thesis.

1.3 Limitations of scope

Although a pedestrian tracking system is developed throughout this thesis, it is important to note that the aim is not to develop a complete system that is suitable for use in a practical scenario. Instead this thesis focuses on the development and evaluation of the system's core components and algorithms, with the aim of answering the research questions posed above. Hence topics such as scaling the system to support many users, location privacy and the distribution of computational tasks over multiple devices are not discussed. Possible avenues of

research in these areas are described in Section 8.2.2. In addition the following limitations of scope also apply:

- This thesis is concerned with the problem of tracking people in an indoor environment. Tracking other objects is not considered, although the algorithms developed in Chapters 5, 6 and 7 can be applied to determine and subsequently track the position of any object that is able to measure its own relative movements to a sufficient accuracy. Algorithms that achieve this requirement for pedestrians are presented in Chapter 4, however they make use of domain specific knowledge and are unsuitable for tracking arbitrary objects.
- The algorithms developed in this thesis assume that walking is the only means by which a pedestrian can move in an indoor environment. This is a valid assumption in most buildings (assuming normal user behaviour), however it is violated by moving floor surfaces such as elevators and escalators. Ways in which the algorithms presented in this thesis could be extended to handle moving floor surfaces are outlined in Section 8.2.1.1, however the implementation and testing of such extensions is left as future work. The assumption is also violated by alternative modes of transportation such as wheelchairs. To support such cases it would be necessary to develop dedicated algorithms to track relative movement for each mode of transportation.

Chapter 2

Positioning systems, Bayesian filters and localisation

A wide range of different sensor technologies have been used to develop indoor location systems. Systems developed in the late 80s and early 90s were nearly all based on a single type of sensor. For example the Active Badge system was based solely on infra-red sensors [5], whilst its successor, the Active Bat system, used only ultrasonic sensors [6]. Other positioning systems were developed based on optical, narrowband and wideband radio technologies. In a summary of such technologies, Welch and Foxlin concluded that a single technology that performs well in all situations has yet to be discovered [17].

The lack of a single *silver bullet* technology for indoor positioning has led to a more recent shift towards the development of indoor positioning systems that use multiple sensor types (an approach that has been used for outdoor positioning since the 1960s and is commonly known as *integrated navigation* [18]). Such systems combine measurements from different sensors in a process known as *sensor fusion*. The idea is to combine measurements in such a way that the resulting system retains the benefits of each individual sensor technology whilst at the same time mitigating their individual weaknesses. One example is the Constellation system, in which both ultrasonic and inertial sensors were used [19].

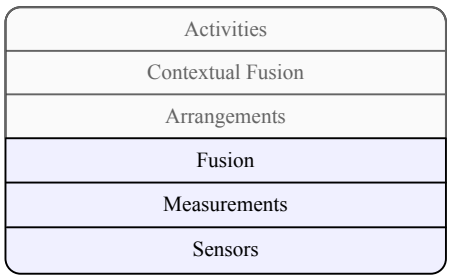


Figure 2.1: The Location Stack model for location-aware computing.

The components of positioning systems that use both single and multiple sensor types can be defined in the context of the bottom three layers of the *Location Stack*, as shown in Figure 2.1. The Location Stack was initially proposed by Hightower et al. as an abstraction for location-aware computing analogous to the Open System Interconnect (OSI) model for computer networks [20, 21]. The upper three layers of the Location Stack abstract services that might be built on top of location systems for use by location-aware applications. The bottom three layers abstract the components of location systems themselves and hence are of direct relevance to this thesis:

- *The sensors layer:* Defines the physical sensors of a location system.
- *The measurements layer:* Defines the types of measurements obtained from the physical sensors.
- *The fusion layer:* Defines the algorithms that combine measurements in order to estimate the positions of tracked objects.

Location systems that use a single sensor type often apply relatively simple deterministic algorithms in their fusion layers in order to estimate the position of a device. For example multi-lateration and multi-angulation algorithms are often used to calculate the position of an object based on multiple distance or angle-of-arrival measurements [22]. In systems that use multiple sensor types, fusion algorithms have the more difficult job of estimating the position of a tracked object based on measurements of different types and with different error characteristics. Furthermore different sensors may have different sampling rates and are often unsynchronized. Bayesian filters provide a mathematical framework that supports sensor fusion in spite of these difficulties [23]. Rather than computing a single *most likely* position estimate, Bayesian filters maintain a probability distribution that describes not only the most likely position of a device, but also the uncertainty in that position given the measurements that have been received. This distribution is evolved over time based on a model of the object's dynamics and is corrected by measurements as they are received, taking the error characteristics of each measurement into account.

The first part of this chapter describes the topics outlined above in more detail. Section 2.1 describes underlying sensor technologies that have been used to develop previous absolute positioning systems. Sensors for tracking the relative position of a device are described in Section 2.2. Bayesian filtering is introduced in its general form in Section 2.3 and its application within positioning systems is described in Section 2.4. The second half of this chapter describes work in the related field of robotics. Section 2.5 describes algorithms that have been developed to solve the robot localisation problem, in which the goal is to allow a robot to determine its absolute position in a known environment using only measurements made by on-board sensors. Finally, the possibility of applying robot localisation techniques to the problem of pedestrian tracking is discussed in Section 2.6.

2.1 Absolute positioning technologies

Many technologies have been used to develop positioning systems for both outdoor and indoor use. Although not directly relevant to this thesis, widely used outdoor positioning technologies are outlined in Section 2.1.1. Technologies that have been developed specifically for indoor positioning are described in Section 2.1.2.

2.1.1 Outdoor positioning

2.1.1.1 Global navigation satellite systems

Global navigation satellite systems (GNSSs) such as the United States' Global Positioning System (GPS) and Russia's GLONASS allow mobile devices to position themselves worldwide when outdoors [4]. To do this a receiver must track signals from four or more satellites, from which pseudo-ranges (distances to satellites that include possible errors due to an offset in the receiver's clock) can be obtained. Pseudo-multi-lateration algorithms (multi-lateration in which the receiver's clock offset is an additional unknown) are then used to resolve the position and clock offset of the receiver. The accuracy to which a GNSS receiver can compute its position is highly dependent on the local environment. With a clear view of the sky and when tracking all visible satellites in the GPS constellation, 1σ position errors of approximately $7\,\mathrm{m}$ can be achieved by single frequency GPS receivers [18]. This accuracy is significantly degraded when large parts of the sky are blocked from view, for example in an urban canyon. In such an environment the number of visible satellites is reduced, whilst multipath effects caused by reflections from nearby obstructions (e.g. buildings and trees) introduce additional error.

In addition to GPS and GLONASS, there are several GNSSs that are currently under development. Europe's Galileo[1] GNSS is due to reach full operational capability between 2016 and 2018. It will be interoperable with GPS and should allow receivers to calculate their positions to within $5\,\mathrm{m}$ horizontally and $8\,\mathrm{m}$ vertically 95% of the time [24]. Galileo satellites will broadcast signals with $5\,\mathrm{dB}$ more power and over wider frequency bands compared to GPS satellites, making their signals easier to detect and more resistant to multipath effects that significantly degrade the accuracy of GPS in urban areas. China is also developing a GNSS called Compass, which is set to provide global coverage by around 2020.

[1] http://ec.europa.eu/transport/galileo

2.1.1.2 WiFi

WiFi access points are now densely deployed in many urban environments. A device can estimate its position by searching a database containing the locations of such access points for ones that are visible from its current location. The accuracy of such an approach is highly dependent on the density of access points, however positioning errors are at worst equal to the range of an access point (typically $50 - 100\,\mathrm{m}$). When multiple access points are visible their signal strengths can be used to estimate a more accurate position. In the Placelab project devices were able to use WiFi access points to position themselves with a median accuracy of $20.5\,\mathrm{m}$ in an urban environment [25].

One benefit of WiFi-based positioning is that the same techniques can be applied in both outdoor and indoor environments. The major disadvantages are that coverage is currently limited to urban environments (since WiFi access points are not yet common in rural areas) and that a database of access points (often called a radio map) must be built for the areas in which such systems are to be used. Furthermore such databases need to be frequently updated as access points are added, moved and removed from the environment. Several commercial companies currently provide WiFi-based positioning solutions. These include Skyhook[2], whose service combines WiFi-based positioning, GPS and position estimation based on signals received from cellular networks.

2.1.1.3 Cellular telephony networks

Cellular network infrastructure for telephony is now widespread throughout large parts of the world. A wide range of measurements can be obtained to estimate the position of any device that is connected to such a network, including angle-of-arrival (AOA), time-of-arrival (TOA) and time-difference-of-arrival (TDOA) measurements of radio signals travelling between the device and multiple cell towers [22, 26]. The simplest approach is to estimate the position of a device as the position of the cell tower to which it is associated, with the accuracy dependent solely on the dimensions of the cell. More accurate approaches have received increased attention since the United States Federal Communications Commission introduced an E-911 mandate that required mobile carriers to provide the approximate positions of mobile devices making emergency calls[3]. By 2012 carriers must provide positions that are accurate to within $50\,\mathrm{m}$ for 67% of calls and within $150\,\mathrm{m}$ for 95% of calls (a deadline that has already been extended by four years from the original deadline of 2008).

It is also possible for a mobile device to estimate its own position based on cell tower signal strengths. This approach is very similar to that commonly used to implement WiFi-based

[2] http://www.skyhookwireless.com

[3] http://www.wirelessdevnet.com/e911/factsheet_requirements_012001.pdf

positioning systems such as those described in Section 2.1.1.2. Cellular networks provide wider coverage, however the accuracy with which a device can position itself using such networks is worse than can typically be achieved using WiFi access points. The Placelab project demonstrated a median accuracy of $107.2\,\mathrm{m}$ when signals from GSM towers were used to position devices in an urban environment [25].

2.1.2 Indoor positioning

Satellite positioning systems such as GPS do not work well (if at all) in indoor environments. To track GPS signals indoors typically requires a receiver capable of tracking signals with power levels ranging from $-160\,\mathrm{dBW}$ to $-200\,\mathrm{dBW}$, however a typical receiver has a noise floor of around $-131\,\mathrm{dBW}$ [27]. Multipath effects are likely to cause a degradation in accuracy even if a receiver is able to track signals from a sufficient number of satellites. Unlike in outdoor environments, reflected signals are often stronger than those received via direct line-of-sight when indoors. Hence it is difficult for a receiver to identify and track the correct (i.e. direct) signals. Although the Galileo system will address some of these problems by transmitting signals at higher power and over wider frequency ranges, its accuracy will not be sufficient to reliably determine the floor of a building on which a device is located. Hence such systems are not accurate enough for indoor location-aware applications, nearly all of which require at least room-level accuracy. Cellular-based positioning systems are similarly inaccurate. WiFi-based positioning systems are able to achieve accuracies sufficient for some location-aware applications. Despite this they have many drawbacks when applied in indoor environments, as will be described in Section 2.1.2.4. As a result other technologies have been developed specifically for indoor positioning, as described below.

2.1.2.1 Infra-red

Infra-red light has been used to provide room-level location information (i.e. the room within which each tracked object is located) [5]. If an infra-red signal transmitted from one device is received by another then it is highly likely that they are located in the same room because infra-red light does not pass through walls. Furthermore infra-red light is strongly reflected by walls, meaning that a line-of-sight between the two devices is not required. Infra-red transmitters and sensors are cheap, power efficient and readily available, making them ideal for providing room-level location information.

The Active Badge system used a network of infra-red receivers to detect badges such as the one shown in Figure 2.2(a), each of which regularly emitted a unique infra-red code [5]. Badges were worn by employees at the Olivetti research laboratory in which the system was deployed. The initial application of the system was to aid receptionists, who were able to

(a) (b)

Figure 2.2: (a) An active badge. (b) An active bat.

route incoming telephone calls to rooms within which their intended recipients were located. After an initial two week trial period it was found that the number of calls that did not reach their intended recipients had dropped substantially.

2.1.2.2 Ultrasound

Ultrasonic positioning systems are able to estimate the positions of tracked objects in three dimensions, with accuracies in the order of centimetres. The Active Bat system positions small narrowband ultrasound transmitters known as bats [28, 6]. Figure 2.2(b) shows an example of a bat designed to be carried by a human user, however this is only one of a number of different designs that were manufactured. To position a bat it is queried over a radio channel. When a bat receives a query it transmits an ultrasonic pulse that is received by multiple ceiling-mounted receivers. The time-of-flight of the signal to each receiver is measured by using the radio query to synchronise the bat and the static receivers. This is possible because the speed of the radio query ($\approx 3 \times 10^8\,\mathrm{ms}^{-1}$) is far greater than the speed of the ultrasonic pulse ($\approx 340\,\mathrm{ms}^{-1}$). When three or more times-of-flight are obtained a multi-lateration algorithm is applied in order to calculate the bat's position. The positions calculated by the system are accurate to within $3\,\mathrm{cm}$ 95% of the time [6].

One problem with the Active Bat system is that ultrasonic noise can prevent a bat from being positioned. Such noise is emitted by a variety of everyday objects including jingling keys and rustling crisp packets. A second problem is that bats in a given area have to be positioned one at a time in order to prevent collisions between their signals. Hence the system does not scale to track large numbers of bats in small areas. The more recent Dolphin positioning system makes use of broadband ultrasound to overcome these limitations [7]. Dolphin is able to position multiple tags simultaneously and in the presence of ultrasonic noise. Its accuracy

is similar to that of the Active Bat system.

The Active Bat and Dolphin systems are both *multilateral* positioning systems, meaning that the position of a tag is calculated by some computational infrastructure in the environment rather than by the tag itself. In contrast Cricket is a *unilateral*[4] ultrasonic positioning system in which mobile tags calculate their own positions by timing ultrasonic pulses that are transmitted by beacons in the environment [8, 29]. To prevent signal collisions beacons in a given area transmit one at a time. This makes positioning a moving tag more difficult than in the multilateral case, since it is not possible to obtain multiple times-of-flight corresponding to the same point in time. As a result the accuracy is significantly worse relative to that of a comparable multilateral system [30]. This problem can be addressed by using broadband ultrasound, as was demonstrated by a prototype system based on the Dolphin hardware [31].

Although unilateral ultrasound positioning systems are less accurate than their multilateral counterparts, they do have several advantages. Multiple tags can calculate their positions based on the same beacon transmissions, meaning such systems scale well to track large numbers of tags in small areas. A second benefit is location privacy, since the only object that can calculate the position of a tag is the tag itself (which can then choose whether or not to share its position).

The main drawback that affects all ultrasound-based positioning systems is that they require large amounts of infrastructure (e.g. receivers or beacons, data cables, power cables and so on) to be installed into the environment. Each tag must have a line-of-sight to at least three receivers (or beacons in a unilateral system) if it is to be successfully positioned. If a user wears a tag around their neck then their body will typically occlude half of the receivers in a given room, meaning that in practice more than three receivers need to be installed in each room in order to achieve acceptable coverage. For example the current Active Bat deployment in the William Gates building (Cambridge) uses 12 receivers to cover a small ($4.5\,\mathrm{m} \times 3\,\mathrm{m}$) office. The number of receivers that are needed could be reduced by requiring that tags are mounted on the users' heads (e.g. attached to hats or helmets), however this would be less acceptable to most users.

2.1.2.3 Ultra-wideband radio

Ultra-wideband (UWB) radio has been used to develop several commercial indoor positioning systems. Ubisense[5] develop systems that are able to position small tags that emit pulses of UWB radio to within $0.13\,\mathrm{m}$ of their true positions 95% of the time in unobstructed line-of-sight conditions (i.e. when there are unobstructed lines-of-sight between tags and receivers).

[4]The Cricket hardware can be deployed to form both multilateral and unilateral positioning systems, however it was originally developed and deployed as a unilateral system.

[5]http://www.ubisense.net

This accuracy degrades to nearer $1\,\mathrm{m}$ when some of the signals have passed through walls. The Ubisense positioning system uses both TDOA and AOA measurements to position tags. TimeDomain[6] have developed a similar UWB positioning system called PLUS in which only TDOA measurements are used.

The main benefit of UWB is that by utilising a wide frequency range it is possible for tags to transmit pulses that are extremely short in duration, making it easier to detect and discard reflected signals [22]. A secondary benefit is that the transmitted signals are able to pass through a wide range of materials, meaning that visible line-of-sight is not required between mobile tags and static receivers (although better accuracies are usually obtained when this is the case). As a result the number of receivers that must be placed into an environment is lower than would be required for an equivalent ultrasound-based system. Unfortunately UWB positioning systems remain very expensive, in part due to the high precision clocks that are required to measure the time differences between radio signals arriving at different receivers in the environment.

2.1.2.4 WiFi

Positioning using received signal strength indication (RSSI) measurements of signals received from WiFi access points has received much attention due to the number of access points that are already installed in many indoor environments. Such systems usually fall into one of two basic classes [22]. In the first class signal strengths are used in conjunction with a radio propagation model in order to estimate distances between the receiver and multiple base stations in the environment. Multi-lateration algorithms can then be applied to estimate the receiver's position. The second class requires the construction of a radio map, in which surveyed signal strengths are recorded at known positions throughout the environment. During tracking a receiver can estimate its position by comparing observed signal strengths with entries in the map. This is known as fingerprinting and is the more accurate of the two approaches, however requires more effort because it is necessary to construct and subsequently maintain a radio map of the environment. In addition to the two classes already described, it is possible to define a third class of system in which a radio propagation model is used to derive some or all of the entries in a radio map.

Radio propagation in indoor environments is dependent on many factors such as the wall and floor materials, which are difficult to model accurately. Complex propagation models have been devised that attempt to take such factors into account [32], however RF-location systems that rely on such models (either to estimate distances between a device and multiple base stations or to derive a radio map) have been unable to match the accuracies achieved by

[6] http://www.timedomain.com

systems in which manually surveyed radio maps are used [10].

A number of different algorithms exist for estimating the position of a device based on a scan of visible access points and a pre-constructed radio map. The RADAR system calculates the Euclidean distance between the vector of signal strengths observed by the device and each entry in the radio map [10]. The position of the device is then estimated as the position in the radio map for which the smallest distance was calculated. This approach results in an error of less than $9\,\mathrm{m}$ 95% of the time. Wang et al. [11] used a similar algorithm to obtain a mean error of $6.44\,\mathrm{m}$. Both of these systems are deterministic, meaning that they compute a single best guess of the position of a device. An alternative approach is to compute a probability distribution of the device's position. The Horus system uses such an approach, with a reported accuracy of $1.4\,\mathrm{m}$ 95% of the time [12].

One of the main problems that limits the accuracy of all WiFi-based positioning systems is that radio propagation is not only dependent on static factors such as building materials, but also on dynamic factors such as whether doors are open or closed and whether humans are present in the environment. The presence of humans in an environment is a particular problem, since a human body obstructing the line-of-sight to an access point causes signal attenuation that can result in a reduction of up to $9\,\mathrm{dBm}$ in the measured RSSI [33]. In practice it is only possible to achieve the kind of accuracies reported by the developers of the Horus system if these dynamic factors are controlled, which is not practical for real-world deployments. A further problem specific to map-based systems is that the manual construction of radio maps is very time consuming [34]. In one of the few large-scale deployments of a WiFi location system, 28 man-hours were required to construct a radio map covering a $12{,}000\,\mathrm{m}^2$ building [35]. Furthermore this process must be repeated whenever an access point is added, moved or removed from the environment if a degradation in the accuracy of the system is to be avoided. The use of radio propagation models to derive some or all of the entries in a radio map can reduce the time required, however the accuracy of the resulting system will also be reduced.

Although the initial interest in WiFi-based positioning was in part due to the fact that access points were already deployed in many buildings, all of the systems described above were tested using a density of access points far higher than would have been installed for the sole purpose of providing wireless connectivity. This is because multiple access points need to be visible to position a device to a reasonable level of accuracy, whilst only one access point needs to be visible for communication. Hence a single access point is sufficient to provide wireless connectivity throughout a small building (e.g. a typical house), whilst at least three are required in order to achieve anything close to the positioning accuracies quoted above.

2.1.2.5 Vision

A wide range of indoor positioning systems have been developed in which cameras are used to track objects in the environment. Tracking pedestrians using multiple camera views is a well researched problem, with techniques such as background subtraction and blob detection commonly used [36, 37]. The EasyLiving project used such algorithms together with two stereo cameras (i.e. four views in total) in order to track the users of an intelligent environment to an accuracy of around $10\,\mathrm{cm}$ in the horizontal plane [38]. This was achieved without requiring a user to wear a tracking device or visual marker. Although this is a highly desirable property, it makes the identification of each user far more difficult. EasyLiving maintained colour histograms of each user to ensure consistent assignment of unique identifiers to users during tracking, however these identifiers did not relate to the absolute identities of the users. A user previously tracked by the system would be assigned a new identifier when re-entering the environment after a sufficiently long absence. Camera-based tracking systems such as the one used in the EasyLiving project often perform poorly when users are partially or fully occluded and also when there are rapid changes in illumination (e.g. when a light is turned on or off, or when clouds obscure the sun on a sunny day).

Camera-based positioning systems span a wide range of accuracies and deployment costs. At the other extreme to the EasyLiving project are expensive motion capture systems, in which multiple cameras are used to track small passive or active markers. Commercial motion capture systems such as Vicon[7] MX can track passive markers to sub-millimetre accuracies at rates in excess of $500\,\mathrm{Hz}$. The markers themselves are designed to be easy to detect, however marker identification is a difficult problem. The IMPULSE motion capture system developed by Phasespace[8] uses active LED markers, which are modulated at unique frequencies to allow identification. Vicon MX allows an operator to define constraints that exist between different markers (e.g. the layout of markers that are attached to a rigid object). When possible this model is used to automatically infer the identity of each marker during tracking.

2.1.2.6 Summary

Table 2.1 summarises the indoor positioning systems described above, although this is by no means an exhaustive list. Other technologies that have been used for indoor positioning include Bluetooth [39], audible sound [40] and signals injected into the power lines running through indoor environments [41]. One property that can be observed for all indoor positioning systems is a strong correlation between accuracy and the amount of infrastructure that must be installed into the environment, as illustrated by Figure 2.3. Whilst ultrasound and

[7] http://www.vicon.com
[8] http://www.phasespace.com

	Location Stack layers					
	Sensor technology	Measurement type	Fusion algorithm	Accuracy range	Infrastructure requirements	Comments
Active badge	Infra-red	Containment	None	Room-level containment	Low	
Active bat	Ultrasound	Time-of-flight	Multi-lateration	2-10 cm (3D)	High	Susceptible to reflections and ultrasonic noise.
Cricket			Various			
Dolphin			Multi-lateration			Broadband ultrasound is more resistant to noise.
Ubisense	UWB	AOA & TDOA	Proprietary	0.1-1 m (3D)	Medium	UWB technology is currently very expensive.
RADAR	WiFi	RSSI	Comparison with radio map	2-10 m (2D)	Medium	Building radio maps is very time consuming. Accuracy degrades as radio environment changes.
Horus			Probabilistic comparison with radio map			
EasyLiving	Vision (markerless)	Image data	Background subtraction & blob detection	0.1-0.5 m (2D)	Medium	Unique identification of tracked objects is difficult.
Vicon MX	Vision (markers)		Proprietary	0.5-5 mm (3D)	Very high	Very expensive. Only covers small areas.

Table 2.1: A summary of indoor absolute positioning systems.

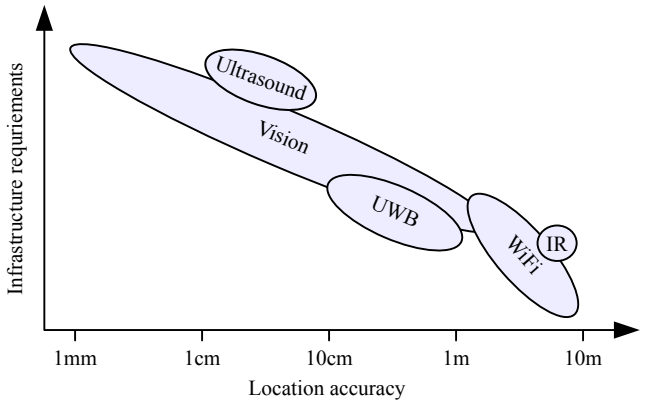


Figure 2.3: The correlation between accuracy and infrastructure requirements for indoor positioning technologies.

vision-based systems are able to achieve very high accuracies, they also have huge infrastructure requirements. Since the amount of infrastructure is correlated to the cost of the system's hardware, it is often prohibitively expensive to deploy accurate positioning systems over large areas. More importantly, the installation and maintenance costs of such systems are also high. These costs (in particular the time required for installation) cannot be expected to fall simply due to increased production levels and improved manufacturing techniques. Hence it is still highly desirable to develop new indoor positioning systems that are located further towards the bottom left of Figure 2.3.

2.2 Relative positioning technologies

Relative positioning systems (also known as *dead reckoning* systems) track the positions of objects relative to their initial locations and orientations. As a simple example consider a pedestrian who counts his steps and remembers any changes in his direction. The pedestrian can use this information together with an estimate of his average stride length in order to track his approximate position relative to his starting point. Note that over time the position that he estimates will become less accurate. This is because he is concatenating estimates of his relative movements (corresponding to steps), each of which will contain an error due to variations in his actual stride length, errors in his estimation of heading changes and so on. These errors accumulate in the estimated position, causing it to become less accurate the further he walks. This accumulation of error is commonly known as *drift* and is common to all relative positioning systems, however the rate at which drift grows and the magnitudes of the errors incurred vary significantly between systems. For some applications drift is acceptable because the target need only be tracked for a short period of time. For others it may be possible to periodically reset the position based on measurements obtained from an absolute positioning system.

2.2.1 Inertial navigation

Inertial navigation is a relative positioning technique in which an inertial measurement unit (IMU) is tracked relative to its initial starting point, orientation and velocity. An IMU that can be tracked in 3-dimensions usually consists of three orthogonal accelerometers and three gyroscopes that are aligned with the accelerometers, as shown in Figure 2.4(a) [18]. An inertial navigation system (INS) consists of an IMU together with a navigation processor, which uses measurements from the IMU in order to track its orientation, position and velocity in some frame of reference in which they are desired (the *reference* frame). The processor uses angular rate measurements obtained from the gyroscopes in order to track the orientation of the IMU relative to this frame. The known orientation is then used to transform the specific force (acceleration due to all forces except for gravity) measured by the accelerometers into this frame. Acceleration due to gravity is then added to the specific force to obtain the acceleration of the device. Finally, this acceleration is integrated once to track velocity and once more to track the device's position in the reference frame.

The type of INS described above is a *strapdown* system, in which the accelerometers measure specific force in the IMU's own frame of reference (the *body* frame). In a *stable platform* INS the accelerometers and gyroscopes are instead mounted on a platform that is held in physical alignment with the reference frame. This is achieved by mounting the platform using gimbals (frames) that allow it freedom in all three axes, as shown in Figure 2.4(b). The gyroscopes are used to detect rotations made by the platform relative to the reference frame. Signals are fed directly from the gyroscopes to torque motors, which rotate the gimbals in order to counteract the rotation and hence keep the platform correctly aligned. Thus acceleration due to gravity can be added directly to the output of the accelerometers in order to obtain the acceleration of the device in the reference frame, which can then be double integrated to track position. The orientation of a stable platform INS relative to the reference frame can be obtained using angle pick-offs that measure the angles between the gimbals of the device.

Stable platform and strapdown INSs are both based on the principle of double integrating acceleration in the reference frame. Strapdown systems have reduced mechanical complexity and as a result tend to be physically smaller and lighter than stable platform systems. These benefits are achieved at the cost of increased computational complexity, however the requirements are now trivial relative to the computational power of modern processors. As a result strapdown systems have become the dominant type of INS.

Inertial navigation makes no assumptions about the movement of the tracked device, except that its accelerations and angular velocities remain within the operating range of the individual sensors. The cost of this generality is that drift in position tends to grow at a rate that is proportional to t^3, where t is the duration for which the device has been tracked [14]. This cubic error growth is due to the propagation of gyroscope bias errors through the computations

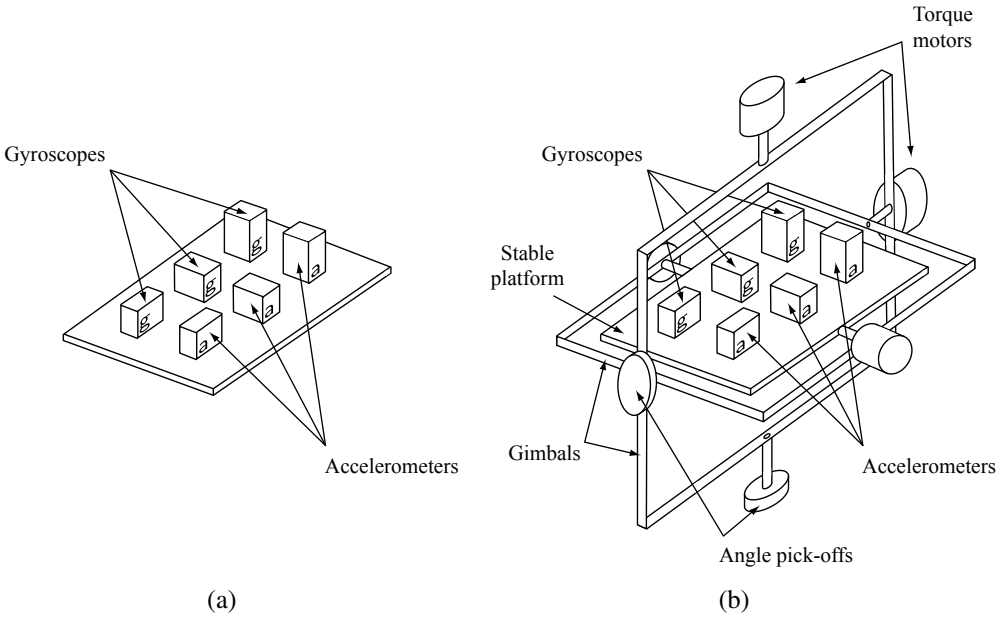


Figure 2.4: (a) A strapdown IMU. (b) A stable platform INS.

performed by the navigation processor, and is described in more detail in Section 3.3. The average error that can be expected after a given duration depends on the quality of the IMU that is used. Foxlin categorises IMUs into several grades, which broadly correspond to different inertial sensor technologies [42]. Figure 2.5 gives lower bounds on the growth of drift for different grades of device. The results shown were obtained by a simulation that considered only random error sources and assumed a stationary device [42]. Note that in practice drift will grow more rapidly due to additional systematic and dynamic errors.

Although a description of the different inertial sensor technologies is beyond the scope of this thesis (such descriptions can be found in [13]) it is worth noting that the sensors required for navigation-grade and strategic-grade IMUs are currently far too large and heavy to attach to a pedestrian. Such devices are typically used in missiles, aircraft and submarines. Note that even a perfect IMU is not by itself sufficient to track position indefinitely without the accumulation of drift. This is because local gravitational conditions continually fluctuate due to tides, seismic activity and the minute gravitational pull of nearby objects. The geophysical limit shown in Figure 2.5 is the growth of drift when using a perfect IMU but without accounting for such fluctuations.

Until relatively recently it was not possible to manufacture IMUs that were small and light enough to use for pedestrian tracking, however recent advances in the construction of micro-machined electromechanical systems (MEMS) have now made it possible to manufacture small inertial sensor chips from which such devices can be built. For example the Xsens[9] Mtx shown in Figure 2.6 is $38 \times 53 \times 21\,\mathrm{mm}$, weighs $30\,\mathrm{g}$ and reports its acceleration and angular

[9] http://www.xsens.com

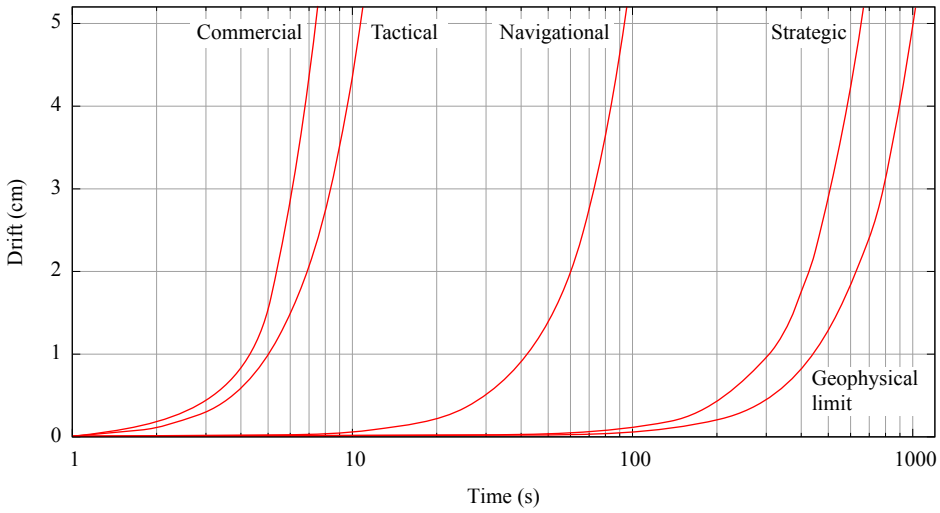
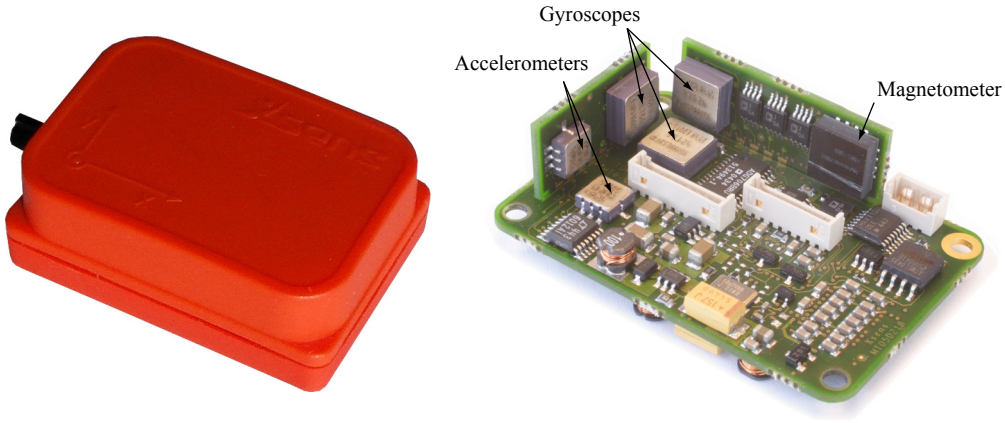


Figure 2.5: Approximate drift due to random error sources when using a commercial-grade, tactical-grade, navigation-grade, strategic-grade or 'perfect' IMU. This graph is based on a graph of simulation results that is presented in [42].

velocity at frequencies as high as $512\,\mathrm{Hz}$. Similar products are available from a number of other manufacturers including Intersense[10] and MemSense[11].

In terms of quality, it is not yet possible to manufacture MEMS IMUs that are of sufficient quality to track the position of a pedestrian to metre-level accuracy for minutes at a time using (uncorrected) inertial navigation. MEMS IMUs are available at tactical-grade performance levels (e.g. the Honeywell[12] HG1900 series), however such devices currently cost several thousands of pounds. They are however rapidly decreasing in price, partly due to the increased use of MEMS inertial sensors in commodity hardware such as mobile telephones and interactive entertainment systems such as the Nintendo[13] Wii. The Xsens Mtx that is used in this thesis currently retails for around £1200 (GBP), however much of this cost can be attributed to additional features such as the inclusion of magnetometers and an embedded Bayesian filter that estimates the absolute orientation of the device. A minimal MEMS IMU of the same quality would currently retail at between £250 and £350. Note that the Xsens Mtx and similar products available from Intersense and MemSense are examples of calibrated IMU packages. It is also possible to purchase uncalibrated MEMS sensors (often sold as separate gyroscope and accelerometer chips). These are typically an order of magnitude cheaper, but also an order of magnitude worse in terms of performance.

[10] http://www.intersense.com
[11] http://www.memsense.com
[12] http://www.honeywell.com
[13] http://www.nintendo.com



(a) (b)

Figure 2.6: (a) The Xsens Mtx IMU. (a) Its internal circuitry (image provided by Xsens).

2.2.2 Pedestrian dead reckoning

Although it is not possible to track the relative movements of a pedestrian (at least to any reasonable level of accuracy) using general purpose inertial navigation, it is possible to do so by using more specialised algorithms. Drift errors grow less rapidly in such algorithms due to their use of domain specific knowledge (e.g. using the known shape of the human gait cycle to detect steps taken by the user). Previous research on pedestrian dead reckoning (PDR) has focused mainly on the problem of tracking emergency and military personnel after they have entered a building. It is not possible to use absolute positioning technologies in such scenarios, since the necessary infrastructure cannot be deployed within the building in advance. Some attempts have been made to develop absolute positioning systems that can be rapidly deployed in such scenarios, for example the Precision Personnel Locator [43], however these have yet to be proven in real environments. Such systems also require the rapid deployment of carefully positioned beacons around the building prior to entry, which may not always be feasible.

Existing PDR algorithms can be split into two main groups: those based on step detection and those based on (constrained) inertial navigation. Both use data from body-mounted MEMS inertial sensors to estimate the relative movements of the user. Step-based algorithms can use sensors mounted in a variety of positions on the body, as described in Section 2.2.2.1. In contrast those based on inertial navigation require foot-mounted sensors in order to be effective, as described in Section 2.2.2.2. Note that in most previous literature the term "pedestrian dead reckoning" has been used to refer exclusively to step-based algorithms, however at a fundamental level this term simply means "the tracking of a pedestrian's relative position". Hence in this thesis the term is used to refer to both step-based systems and those based on inertial navigation.

2.2.2.1 Step-based PDR

Step-based PDR algorithms consist of three phases: step detection, step length estimation and step heading estimation. Each time a step is detected its estimated length and heading are used to update the position of the tracked user. Step detection can be done using an accelerometer mounted on the subject's waist [44], foot [45], backpack [15] or even helmet [46]. For example signals from waist-mounted accelerometers exhibit a double-peaked oscillatory pattern during walking [47]. This allows steps to be identified using simple techniques such as the detection of zero-crossings [15] or peaks [48] in the signals. Step detection using a foot-mounted accelerometer is even easier, since it is trivial to determine when the foot is in contact with the ground by detecting stationary regions in the signals.

Estimating the length of each step is more difficult. The simplest approach is to use a fixed step length that can be set for each user. This approach will give a reasonable level of performance when the user is walking at a constant speed, but will perform poorly otherwise. This is because step lengths for a given person can vary by up to $\pm 50\%$ depending on walking speed, with a typical person taking smaller steps when walking slowly compared to when they walk at a brisk pace [49]. Several more advanced algorithms have been developed for estimating step lengths [11, 44]. For example Alvarez et al. proposed the estimator

$$l = \frac{k_1}{t_{\text{step}}} + k_2 \tag{2.1}$$

where l is the estimated step length, t_{step} is the measured step duration and k_1 and k_2 are constants that must be determined on a per-user basis [44]. Neural networks have also been used to predict step lengths based on feature vectors extracted from accelerations that are measured during each step [46]. This approach was shown to estimate step lengths to within a few percent, however it is necessary to train a separate neural network for each user. This requires acceleration data to be obtained for which the corresponding step lengths are known, which is not always practical and can be time consuming.

Step headings can be estimated by using a magnetometer to measure the user's absolute heading, which can then be taken as the direction of the step. One problem with this approach is that it assumes that the user is always walking forwards. Hence an error is introduced if the user steps sideways or backwards. Magnetometers are also affected by magnetic disturbances which in some cases render their measurements useless [50]. Some systems use gyroscopes in addition to magnetometers in order to detect and compensate for such disturbances [46]. Others use only gyroscopes, however such systems are only able to measure relative changes in heading over successive steps (which causes drift to accumulate in the tracked orientation as well as in the position). All of these approaches usually require sensors to be rigidly attached to the user. When magnetometers are used to calculate absolute headings, the orientation be-

tween the sensor and the user's body must also be known. Recent research has attempted to relax these constraints, however the accuracies of the resulting systems have been significantly worse than those in which fixed mounting points are used [48].

PDR systems based on step detection have the advantage that they can operate with low grade inertial sensors such as those found in many mobile telephones. Unfortunately step-based PDR systems also have a number of drawbacks. They must be calibrated for each user and most require a sensor to be rigidly mounted (and in some cases correctly aligned) with the user's body in order to obtain accurate heading measurements. Well calibrated step-based PDR systems accumulate drift that is in the order of 3% of the distance travelled, however this is only achievable if the user does not make any unusual movements such as walking backwards or to the side [51]. Such movements violate the assumptions made by the underlying algorithms and hence introduce errors. One final problem that is of particular note for indoor tracking is that most step-based PDR systems are unable to detect changes in vertical displacement, for example when a user ascends or descends a flight of stairs.

2.2.2.2 Inertial navigation based PDR

In a PDR system based on inertial navigation techniques a full INS is used to track the position of the user. The rapid accumulation of drift in the tracked position is prevented by frequently correcting the INS. This is achieved by placing the IMU on the user's foot, which has a known velocity of zero whenever the foot is grounded. Furthermore it is straightforward to identify periods during which the foot is grounded, since the acceleration and angular velocity signals reported by the device during such periods are near-static. This allows zero velocity updates (ZVUs) to be applied in order to correct the INS [14, 16]. As well as correcting any errors that have accumulated in the tracked velocity, the application of ZVUs can also be used to correct tilt errors in the tracked orientation and partially correct drift errors that have accumulated in the tracked position during the previous step. This is possible due to correlations that exist between accumulating errors in different components of the INS state [14]. Bayesian filters provide an ideal framework to track such correlations and to apply ZVUs in a way that allows correlated errors in all components of the state of the INS to be corrected. In particular extended Kalman filters have been widely used [15, 14, 16].

INS-based PDR systems have both advantages and disadvantages compared to step-based systems. Their biggest advantage is that they make far fewer assumptions about the user's motion. In particular they do not assume that the user is walking forwards. Hence such systems correctly handle unusual movements such as sidesteps and walking backwards. The user's vertical displacement is also tracked correctly when he or she ascends or descends a flight of stairs. INS-based PDR systems tend to be more accurate than step-based systems. For example the NavShoe system (an INS-based system) was shown to accumulate drift in

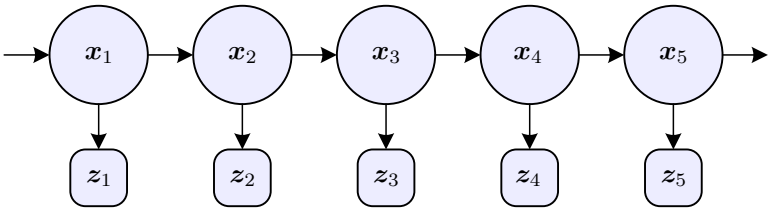


Figure 2.7: A Markov process.

the order of 0.3% of the total distance travelled [14] (albeit with the additional use of magnetometers to correct heading errors), compared to around 3% achieved by good step-based systems. The main disadvantage of INS-based systems is that they always require a full 3-dimensional IMU. In contrast it is possible to construct a step-based system using only a single axis accelerometer together with a single axis magnetometer. Hence step-based systems can be significantly cheaper. Another disadvantage of INS-based systems is that the IMU must be mounted on the foot, which is less acceptable to most users compared to a mounting point on the waist.

2.3 Bayesian filters

The inherent uncertainty of the raw measurements obtained from sensors, together with the desire to use multiple sensor types, has led to the widespread use of Bayesian filters in positioning systems. Bayesian filters provide a framework for probabilistically estimating the state of dynamic systems based on noisy measurements [52]. For pedestrian tracking the dynamic system is (in some sense) the pedestrian and its state is the pedestrian's position along with any additional information such as their heading or velocity. Bayesian filters are deployed in the fusion layer of the Location Stack model of positioning systems shown in Figure 2.1, combining measurements obtained from individual sensors to form an estimate of the user's position [21]. In this section Bayesian filters are introduced in their general form. The use of Bayesian filters in positioning systems is described in Section 2.4.

A Bayesian filter represents the state of a dynamic system at time t as a probability distribution over the space of all possible states

$$\mathrm{Bel}(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|\boldsymbol{z}_1, ..., \boldsymbol{z}_t) \tag{2.2}$$

where $\boldsymbol{z}_1, ..., \boldsymbol{z}_t$ are noisy measurements made up to (and including) time t. Representing the state as a probability distribution allows a Bayesian filter to provide both an estimate of the true state of the system and a measurement of the uncertainty in that estimate. More generally, the belief $\mathrm{Bel}(\boldsymbol{x}_t)$ can be queried to obtain the probability that a state $\boldsymbol{x}_t$ is the true state of the system, given all of the measurements made up until time t.

In general the complexity of computing $\mathrm{Bel}(\boldsymbol{x}_t)$ increases exponentially as the number of measurements increases [23]. To make the computation tractable Bayesian filters assume that the dynamic system is a Markov process, as shown in Figure 2.7, in which the next state depends only on the current state and in which a measurement made at time t is dependent only on the state $\boldsymbol{x}_t$ at that time. Such systems are defined by two distributions: a propagation model $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-\delta t})$ and a measurement model $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$. The propagation model defines how the system's state changes over time, whilst the measurement model defines the probability of making an observation $\boldsymbol{z}_t$ given the current state $\boldsymbol{x}_t$.

Under the Markov assumption $\mathrm{Bel}(\boldsymbol{x}_t)$ can be calculated by updating the previous distribution $\mathrm{Bel}(\boldsymbol{x}_{t-\delta t})$, allowing a Bayesian filter to track the state of a dynamic system through time. To perform the update the belief is propagated forward according to the system's propagation model to obtain the estimated belief, or *prior* distribution

$$\mathrm{Bel}^{-}(\boldsymbol{x}_t) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-\delta t})\mathrm{Bel}(\boldsymbol{x}_{t-\delta t})\, d\boldsymbol{x}_{t-\delta t}\,. \tag{2.3}$$

The prior is then corrected using the measurement $\boldsymbol{z}_t$ to obtain the new belief, or *posterior* distribution

$$\mathrm{Bel}(\boldsymbol{x}_t) = \alpha_t p(\boldsymbol{z}_t|\boldsymbol{x}_t)\mathrm{Bel}^{-}(\boldsymbol{x}_t) \tag{2.4}$$

where α_t is a normalisation factor. In some cases it is necessary to update the state even when no new measurements are available. In this case the prior is calculated and is taken as the new posterior distribution. It is also possible to apply multiple measurements in a single update by using the previously calculated posterior as the prior in successive iterations of the correction step. Different measurement models can be used to incorporate different measurement types. Hence Bayesian filters naturally support sensor fusion, in which measurements of different types are combined to form an overall estimate of the state.

Bayesian filters provide a framework for probabilistically estimating the state of a dynamic system, however their definition does not specify how such a filter should be implemented. In particular a concrete implementation must choose how to represent the belief $\mathrm{Bel}(\boldsymbol{x}_t)$. A range of different implementations have been developed with varying computational requirements and underlying assumptions. The most widely used types of Bayesian filter are described below.

2.3.1 Kalman filters

Kalman filters are the most widely used variant of Bayesian filters due to their computational efficiency. They assume that $\mathrm{Bel}(\boldsymbol{x}_t)$ is Gaussian distributed for all t [53]. Hence the belief can be represented simply by a mean and covariance. It is also assumed that the state of the

system evolves according to a linear equation of the form

$$\boldsymbol{x}_t = \boldsymbol{F}_t \boldsymbol{x}_{t-\delta t} + \boldsymbol{w}_t \tag{2.5}$$

where $\boldsymbol{F}_t$ is a state transition matrix that defines how the state evolves over time and $\boldsymbol{w}_t = \mathcal{N}_{0,\boldsymbol{Q}_t}$ is zero mean Gaussian distributed noise with a known covariance $\boldsymbol{Q}_t$. Each measurement must be related to the state by a linear function of the form

$$\boldsymbol{z}_t = \boldsymbol{H}_t \boldsymbol{x}_t + \boldsymbol{v}_t \tag{2.6}$$

where $\boldsymbol{H}_t$ is a matrix that defines the relationship between the measurement $\boldsymbol{z}_t$ and the current state. The measurement noise $\boldsymbol{v}_t = \mathcal{N}_{0,\boldsymbol{R}_t}$ is assumed to be zero mean Gaussian distributed with a known covariance $\boldsymbol{R}_t$. Kalman filters are optimal Bayesian filters given these assumptions, meaning that they compute Equations 2.3 and 2.4 exactly [54]. Furthermore, they do so using computationally efficient matrix operations (given in Appendix C). Hence Kalman filters are the preferred type of Bayesian filter for linear or near-linear systems where the belief will always be uni-modal. Their computational efficiency makes it possible to usefully deploy Kalman filters on embedded and mobile devices.

2.3.2 Linearised and extended Kalman filters

Kalman filters as described above can only be applied to near-linear systems and measurements. Linearised and extended Kalman filters (EKFs) relax these constraints by linearising the more general propagation and measurement functions

$$\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_{t-\delta t}) + \boldsymbol{w}_t \tag{2.7}$$

$$\boldsymbol{z}_t = \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t) + \boldsymbol{v}_t \tag{2.8}$$

using first-order Taylor series expansions [53], where $\boldsymbol{\mathcal{F}}_t$ and $\boldsymbol{\mathcal{H}}_t$ are differentiable (but possibly non-linear) functions. The linearised Kalman filter performs the linearisation about an approximate state trajectory that must be known in advance. For example the intended flight path of a missile aimed at a stationary target could be used as an approximate trajectory when tracking the missile's true position.

The EKF is similar to the linearised Kalman filter except that the linearisation takes place about the filter's estimated trajectory rather than a pre-computed one. Hence it is suitable for problems where an approximate trajectory is not known in advance, as is often the case. Their main disadvantages are that they are still only able to represent Gaussian belief and that they are more likely to diverge from the true state in unusual situations than both regular and linearised Kalman filters [53].

2.3.3 Multi-hypothesis filters

A Multi-hypothesis filter consists of multiple Kalman filters (or EKFs), each of which tracks one of a number of possible hypotheses. The overall belief is a mixture of the Gaussian distributions maintained by the individual Kalman filters, which are weighted according to how closely they predict the observed measurements [23]. The main benefit of using a multi-hypothesis filter is that it is possible to represent multi-modal distributions. The main drawback is increased computational complexity. It is also necessary to implement heuristic algorithms in order to determine when to remove or introduce hypotheses. Finally, note that the tracking of each individual hypothesis is still subject to the restrictions and assumptions that are imposed by the type of Kalman filter used (e.g. zero mean Gaussian distributed noise).

2.3.4 Grid-based filters

Grid-based filters are optimal Bayesian filters for systems in which the state space is both discrete and finite [54]. Given this assumption the belief can be represented by a discrete set of weights $\{w_t^i \mid i = 1, ..., N\}$, where w_t^i is the probability of state $\boldsymbol{x}_t^i$ and $\{\boldsymbol{x}_t^i \mid i = 1, ..., N\}$ is the set of possible states at time t. The belief is given by

$$Bel(\boldsymbol{x}_t) = \sum_{i=1}^{N} w_t^i \, \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^i) \tag{2.9}$$

where $\delta(x)$ is the Dirac delta function. No assumptions are made about the form of the propagation and measurement models $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-\delta t})$ and $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$.

Grid-based filters can also be applied to systems in which the state space is continuous (but still finite) by dividing the state space into N discrete cells $\{\boldsymbol{x}_t^i \mid i = 1, ..., N\}$. The weight w_t^i is then the probability that the true state of the system lies within the cell $\boldsymbol{x}_t^i$ at time t.

The main advantages of grid-based filters are that they can represent arbitrary probability distributions across the set of discrete states and that they do not place any restrictions on the types of propagation and measurement models that can be used. The main disadvantages are that they require $\mathcal{O}(N)$ memory and to update the state requires $\mathcal{O}(N^2)$ computation (since each of the N weights in the new state is calculated by a summation over all N states [54]). Hence they do not scale well to large state spaces. In practice this means that grid-based filters are restricted to problems of low dimensionality, since the number of states in a space normally grows exponentially with the number of dimensions. When a continuous state space is discretised in order to apply a grid-based filter, there is a trade-off between using larger cells to reduce computational overheads and using smaller cells to minimise discretisation errors that are introduced as a result.

2.3.5 Particle filters

Particle filters approximate the belief as a set of weighted samples, or *particles*

$$\mathcal{S}_t = \{\langle \boldsymbol{x}_t^i, w_t^i \rangle \;\; i = 1, ..., N\} \tag{2.10}$$

where $\boldsymbol{x}_t^i$ is the state of the i^{th} particle and w_t^i is its weight. The weight of a particle can be thought of as the probability that the particle's state corresponds to the true state of the system. Various algorithms exist for updating the belief, almost all of which are based on the Bootstrap filter [55] (also known as the Sampling-Importance-Resampling (SIR) filter). The Bootstrap filter's update procedure consists of three steps. First, each particle is propagated according to the system's propagation model. Their weights are then adjusted according to any measurements, each of which must have a corresponding measurement model. Finally the particles are re-sampled in proportion to their weights to obtain a new set of equally weighted particles, which approximates the new posterior distribution. The re-sampling process causes more particles to be generated in areas of the state space that are likely to correspond to the true state of the system. Hence unlike grid-based filters, particle filters automatically focus computation where it is most needed.

The main advantage of particle filters is that they do not place any restrictions on the form of the propagation and measurement models. Like grid-based filters they are able to represent arbitrary multi-modal distributions, however unlike grid-based filters they do not require a discrete or finite state space. Hence particle filters are well suited to problems in which the state space is continuous and the belief is multi-modal. A particle filter requires $\mathcal{O}(N)$ memory and to update the state requires $\mathcal{O}(N \log N)$ time (although by using more efficient re-sampling techniques this can be reduced to $\mathcal{O}(N)$ [56]), where N is the number of particles. There exist methods for dynamically varying the number of particles and hence the computational costs [57, 58]. Such algorithms typically generate fewer particles when uncertainty is low relative to when it is high.

2.3.6 Summary

Table 2.2 gives a summary of the types of Bayesian filter described above. When selecting a Bayesian filter it is necessary to trade off the computational requirements of the filter with the restrictions imposed on the belief distribution and the propagation and measurement models. Both regular and extended Kalman filters have been applied to many systems in which belief is uni-modal. They can often be applied successfully in practice to systems that violate their assumptions, for example the Gaussian nature of process and measurement noise. More computationally expensive filters such as particle and grid-based filters are usually only required

when it is necessary to represent more complex multi-modal beliefs.

2.4 Bayesian filters in positioning systems

Bayesian filters have been widely used in the fusion layer of positioning systems. In most cases such filters are used to implement a technique known as Single-Constraint-At-A-Time (SCAAT) tracking [59], in which measurements from individual sensors are incorporated one at a time into the estimated state of the tracked object(s). This approach is fundamentally different from the traditional lateration and angulation algorithms used by many of the positioning systems described in Section 2.1, in which a certain number of measurements must be received (e.g. three in the case of lateration) before the object's position can be calculated. In SCAAT tracking individual measurements can be used to update the estimated state as soon as they are received by the filter. Hence a measurement can be used to update the estimated state of a device even if an insufficient number of measurements are available to fully determine its position. This ability is particularly useful for unilateral ultrasound-based positioning systems such as Cricket, in which mobile tags obtain range measurements to beacons in the environment one at a time. Cricket tags are not able to simultaneously measure the range to multiple beacons, making the application of traditional lateration algorithms difficult and inaccurate. In contrast a Bayesian filter can use SCAAT tracking to incorporate the individual range measurements into an estimate of the tag's true position, updating the estimate whenever a new measurement is received [30].

The HiBall tracker is another example of a positioning system in which a SCAAT approach is used [60, 61]. A HiBall consists of a compact vision-based system that detects flashing LEDs mounted in the ceiling of the environment. SCAAT is used to incorporate individual measurements into an estimate of the HiBall's state (i.e. its position and orientation). Using this approach state estimates can be generated at $2000\,\mathrm{Hz}$, with less than $1\,\mathrm{ms}$ latency and with maximum errors in position and orientation that are of the order of $1\,\mathrm{mm}$ and $0.05°$ respectively. The system was designed for virtual reality applications, where rapid updates, low latencies and high accuracies are important.

2.4.1 Sensor fusion

Although the ability to incorporate individual measurements into the estimated state is a major benefit, it is not the main reason why Bayesian filters are so often used in location systems. The most important property of Bayesian filters for location systems is their ability to perform sensor fusion, in which measurements from different types of sensor are combined to form a single estimate of the state of a tracked device. Recall from Section 2.3 that a Bayesian filter

	Belief	Propagation and measurement models	Computational requirements	Implementation difficulty	Comments
Kalman filter	Gaussian	Linear	Low	Low	
Linearised KF	Gaussian	Linear about approximate trajectory			Approximate state trajectory must be known in advance.
EKF	Gaussian	Non-linear			May diverge in unusual situations.
Multi-hypothesis filter	Mixture of Gaussians	Non-linear	Medium	High	Heuristics needed to decide when to add and remove hypotheses.
Grid	Arbitrary (discrete)	Arbitrary (discrete)	Very high	Medium	State space must be finite and discrete. Particle filters generally better unless state space is naturally discrete.
Particle filter	Arbitrary	Arbitrary	High	Low	Can dynamically vary the number of particles as needed.

Table 2.2: A summary of commonly used Bayesian filters.

estimating the time-varying state $\boldsymbol{x}_t$ of a system can incorporate any measurement $\boldsymbol{z}_t$ at time t, provided that a corresponding measurement model $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ is available in a form suitable for the type of Bayesian filter that is being used. Hence Bayesian filters are able to combine measurements from different types of sensor into a single position estimate, provided that measurement models for each type of measurement are available. Each measurement model encodes the accuracy and error characteristics of the underlying sensor, which are taken into account by the Bayesian filter when updates are performed. Note that a SCAAT approach is particularly useful when multiple sensor types are used, since different sensors often generate measurements at different rates and at different points in time.

Sensor fusion is most useful when complementary sensor types are used (i.e. when a weakness of one sensor type is a strength of another). A common example is inertial-GPS, in which a GPS receiver is enhanced by the addition of an IMU. Inertial-GPS devices are able to track their position and orientation outdoors at a high update rate (a benefit provided by the IMU) without incurring drift (a benefit provided by GPS) at an accuracy that is better than can be achieved using GPS alone [13]. Applications include machine control (e.g. farming machinery), surveying and vehicular positioning [62], missile guidance (often in conjunction with radar) [63] and the accurate registration of video captured from moving platforms [64].

Sensor fusion is even more relevant to indoor positioning due to the lack of anything approaching a single sensor technology that is accurate and reliable enough for all situations [17]. The Constellation tracking system uses a Kalman filter to combine measurements from an IMU with range measurements to ultrasonic beacons in the ceiling of the environment [19]. Constellation is able to estimate the position and orientation of the tracked device to within $8\,\mathrm{mm}$ and 1° of its true state. Existing deployments are relatively small (e.g. covering a single room), in part due to the high infrastructure requirements.

An implementation of the Location Stack used sensor fusion for more coarse grained tracking of user's in indoor environments, using particle filters to combine a variety of measurements including ultrasonic range measurements and infra-red proximity measurements [65]. A deployment in a $900\,\mathrm{m}^2$ office was able to position users to within $2\,\mathrm{m}$ 90% of the time, compared to approximately $3\,\mathrm{m}$ and $4\,\mathrm{m}$ when deterministic algorithms were applied to measurements from the individual infra-red and ultrasonic systems respectively.

2.4.2 Constraints as pseudo-measurements

In addition to combining measurements from multiple sensor types, Bayesian filters are also able to factor in constraints that are known to apply to the state of a tracked device. Such constraints are enforced through the application of *pseudo-measurements*; imaginary measurements whose corresponding measurement models describe the nature of the constraint.

For example consider a device whose position and velocity is tracked by a Bayesian filter. If the device is known to have a maximum speed of $5\,\mathrm{ms}^{-1}$ then this can be enforced by applying pseudo-measurements with the measurement model

$$p(\mathrm{MaxSpeed}|\boldsymbol{x}_t) = \begin{cases} 0.2 & |\boldsymbol{v}_t| < 5 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

after each propagation step, where $\boldsymbol{v}_t$ is the velocity component of the state. Note that since pseudo-measurements are treated just like regular measurements in the Bayesian framework, their measurement models must be in a form suitable for the type of Bayesian filter used. For example $\mathrm{MaxSpeed}$ constraints could not be applied by a Kalman filter, since they violate the Kalman filter's linear and Gaussian assumptions. Such constraints could however be applied by a particle or grid-based filter.

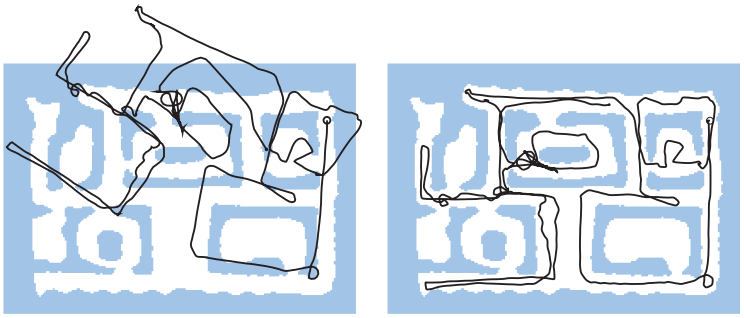
Using pseudo-measurements to enforce constraints has been widely applied within Bayesian implementations of inertial navigation based PDR systems, as described in Section 2.2.2.2. In such systems an IMU attached to the user's foot is known to be stationary whilst the user's foot is grounded. This constraint can be enforced by applying zero velocity pseudo-measurements [14].

2.5 Robot localisation

The indoor absolute positioning systems described in Section 2.1.2 rely on the installation of fixed infrastructure such as ultrasonic beacons and WiFi access points. In some situations it is desirable to track the location of pedestrians in an indoor environment, however it is undesirable to have to install such infrastructure in advance. Sometimes it is impossible to do so, for example when tracking emergency services personnel as they enter a building to which they have just been called. This has led to the development of PDR systems as described in Section 2.2.2, however drift causes the accuracy of such systems to degrade over time. The initial position and orientation of the user must also be known.

One related area of research in which the problems outlined above have been addressed is that of robot localisation. Robot localisation is the problem of determining and subsequently tracking the position and orientation of a robot that is initially placed at an unknown location within a known environment. This must be done using only sensors that are attached to the robot and a map of the environment. The robots used to investigate this problem are typically equipped with laser or ultrasonic range-finders [66, 57], which are used to measure distances to nearby obstructions. Upward facing cameras have also been used [67].

In addition to the sensors described above, the robot's wheels are typically equipped with



(a) (b)

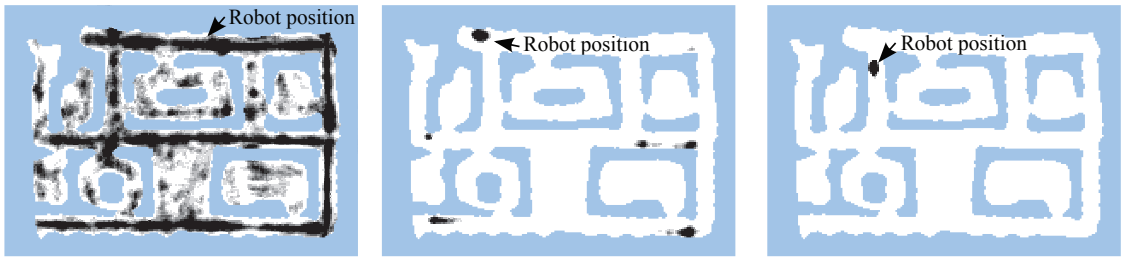
Figure 2.8: (a) The path followed by a mobile robot, as estimated by odometry. (b) The corrected path. Adapted with permission from [66].

sensors (usually optical rotary encoders) that measure wheel rotation. Measurements from such sensors can be used to estimate changes in the robot's position and orientation. The technique of tracking the relative movement of a robot in this way is known as odometry [68]. As with any dead reckoning technique, errors will accumulate in the tracked position over time. Sources of error include wheel slippage, floor roughness and the discretised sampling of wheel rotation [69]. Figure 2.8(a) shows an example of a path estimated by odometry, which has been aligned with the map to have the correct starting position and orientation. The actual path followed by the robot is shown in Figure 2.8(b).

It is possible to determine the absolute position of a robot in a known environment (and hence solve the robot localisation problem) because only some paths through that environment will be consistent with both the measurements made by the robot and its path as estimated by odometry. As the robot senses more of the environment its position can be narrowed down further as more paths are discounted. This process continues until the robot's position has been uniquely determined, as shown in Figure 2.9. Bayesian filters provide an ideal framework for robot localisation, since they are able to represent the changing belief in the robot's position throughout the localisation process. The belief is typically multi-modal and non-Gaussian during localisation, as shown in Figures 2.9(a) and 2.9(b). Grid-based filters and particle filters are both able to represent such distributions and have been successfully applied to the robot localisation problem (e.g. [70] and [71] respectively).

Whichever filter is used, it is necessary to specify a propagation model and a measurement model for each type of measurement. The propagation model $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ updates the belief to account for the movement of the robot since the last update, where $\boldsymbol{u}_t$ is is the robot's estimated relative movement obtained by odometry. For a range measurement r_t in direction θ_t, a common measurement model for $\boldsymbol{z}_t = (r_t, \theta_t)$ is given by

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t) = \mathcal{N}_{0,\sigma}(r_t - \text{range}(\text{map}, \boldsymbol{x}_t, \theta_t)) \tag{2.12}$$



(a) (b) (c)

Figure 2.9: Density plots after incorporating (a) 5, (b) 18 and (c) 24 sonar scans during robot localisation. In this example a particle filter is used to perform the localisation. Darker positions are more likely. Adapted with permission from [66].

where $\mathrm{range}(\mathrm{map}, \boldsymbol{x}_t, \theta_t)$ is the range to the nearest obstruction in the map of the environment from the position defined by $\boldsymbol{x}_t$ in the direction θ_t. The nearest obstruction can be found by ray-tracing [66].

Early solutions to the robot localisation problem used position probability grids (a type of grid-based Bayesian filter), in which the state space of all possible positions and orientations is divided into discrete cells [70, 72]. Spatial and angular resolutions of around $20\,\mathrm{cm}$ and $5°$ were typically used. This approach was shown to allow successful robot localisation and subsequent tracking with a mean position error of $5\,\mathrm{cm}$ for a robot equipped with a laser-range finder [66]. Despite this there were several notable drawbacks. The memory and computational requirements were proportional to the number of cells and hence to the resolution of the grid, with a finer grid requiring more resources. Reducing the resolution of the grid lowered the requirements but also increased discretisation errors, lowering the probability of successful localisation and decreasing the accuracy during tracking.

Pre-computing the sensor model $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ and performing selective updates have both been used as techniques for reducing the computational requirements of grid-based localisation algorithms [66]. Pre-computing the sensor model for every possible cell and measurement combination avoids the need to ray-trace values of $\mathrm{range}(\mathrm{map}, \boldsymbol{x}_t, \theta_t)$ on-line, reducing the computational cost of the algorithm by a constant factor. Selectively updating only cells that are actually likely to correspond to the true position of the robot can drastically reduce the computational requirements of the filter when uncertainty in the robot's position is low. This approach was demonstrated to reduce the computational load by over 95% after a robot's position was determined in a test localisation, since only a few cells corresponded to the robot's known position [66]. The main disadvantage of such an approach is that the complexity of implementing the algorithm is significantly increased.

More recently particle filters have been preferred to grid-based filters for robot localisation. Particle filters do not require a discretisation of the state space and automatically focus computation where needed, since re-sampling causes more particles to be generated in areas where

the robot is likely to be positioned. The number of particles can also be adapted, so that more particles are used when the uncertainty is high relative to when the robot's position has been determined [57, 58]. The effect of adaptive re-sampling is similar to that of selective updates in grid-based filters, however it is far easier to implement. Particle filter implementations have been shown to achieve superior accuracy relative to grid-based approaches, whilst requiring an order of magnitude less computation and memory [71].

Once a robot localisation algorithm has determined the position of the robot, it can continue to track the robot's position indefinitely simply by applying the propagation and measurement models in the same way as during localisation. The relative movement measurements provided by the robot's odometry data are used to propagate the robot's position, whilst the measurements from sensors attached to the robot are used together with the map of the environment in order to prevent drift that would otherwise accumulate in the tracked position. Figure 2.8(b) shows the corrected path calculated by a localisation filter for the robot whose raw odometry data is shown in Figure 2.8(a).

2.5.1 Dealing with unmapped obstructions

One of the major problems when deploying robots in realistic environments is that of dynamic obstructions such as pedestrians. Robot localisation as presented above assumes that any measurement (e.g. a range measurement) is of the mapped environment. Hence a measurement of a dynamic obstruction will be deemed inconsistent with the robot's true state, causing a tracking error or localisation failure.

One solution to this problem is to pre-filter measurements, discarding any that are likely to be of dynamic obstructions. Fox et al. describe filtering techniques based on both entropy and distance [66]. Their entropy filter applies a measurement only if it will result in a reduction in the entropy of the belief (which is a measure of uncertainty). Hence a measurement is only used if it re-enforces the current hypothesis. Their distance filter rejects range measurements that are very likely to be shorter than expected given the current belief. These algorithms were tested using robots deployed in museum environments through which many pedestrians moved. Both reduced the duration for which the tracking error was over a threshold of $45\,\mathrm{cm}$ from 26.8% of the time to just over 1%. The entropy approach has the advantage that it can be applied equally to any measurement type (not just range measurements), however takes longer to recover when a tracking error does occur. It is unclear whether either method would be effective during localisation, since it is much harder to determine whether or not a measurement should be filtered when uncertainty in the robot's position is high.

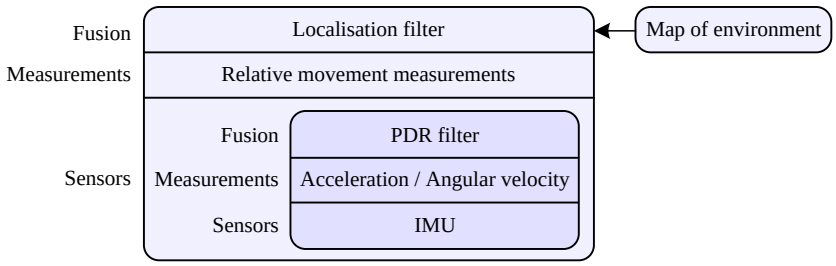


Figure 2.10: Cascaded stacks in a pedestrian localisation system.

2.6 Pedestrian localisation

One of the goals of this thesis is to determine a way in which the absolute position and heading of a person can be determined in an indoor environment, without having to install large amounts of infrastructure. The algorithms outlined in Section 2.5 are able to determine the absolute position and heading of a mobile robot without the need for any infrastructure, provided that a map of the environment is available. It should be possible to develop similar algorithms for pedestrians, combining relative movement measurements obtained from a PDR system with constraints derived from a map of the environment in order to determine and subsequently track a user's absolute position. During tracking the same algorithms could be used to limit drift that would otherwise accumulate in the tracked position.

In the context of the Location Stack representation for positioning systems shown in Figure 2.1, the approach described above can be thought of as embedding the bottom three layers of one stack inside the sensors layer of another higher level stack. The lower level stack corresponds to the PDR system itself, defining the inertial sensors, measurements and algorithms that are used to estimate the relative movements of the pedestrian. This stack is then treated as a sensor in the sensors layer of the higher level stack, which combines the measurements provided by the PDR system with a map of the environment as shown in Figure 2.10.

2.6.1 Differences to robot localisation

There are many similarities between the pedestrian localisation problem and the robot localisation problem described in Section 2.5. There are also several important differences. Firstly, existing solutions to the robot localisation problem have only considered environments consisting of a single floor. Supporting environments with multiple floors was not necessary since the robots that were used to test such implementations were unable to climb stairs. The extra mobility of a pedestrian makes it necessary to support such environments.

Secondly, existing robot localisation systems use robots that provide both relative movement measurements (usually obtained using wheel encoders) and measurements from additional

sensors (usually range measurements to nearby obstructions obtained using laser or ultrasonic range finders). For pedestrian localisation it is desirable to keep the number of sensors to a minimum so as to minimise the inconvenience to the tracked user. Furthermore the lack of rigid mounting points on the human body makes it difficult to use sensors that are typically used for robot localisation. In this thesis a solution to the pedestrian localisation problem is presented that uses only a small IMU to measure the approximate relative movement of the user. In effect the user is used as a sensor that detects the *absence* of obstacles such as walls in the environment (since a person cannot pass through such obstacles). This task is significantly more difficult because less information is available. The error characteristics of the positions calculated by the resulting system are significantly different to those of a typical robot localisation system, as will be described in Chapter 6.

A final but important difference is that a pedestrian localisation system cannot exert control over the movement of the user. A robot localisation system can direct a robot to move in a way that helps the localisation process. For example if there are two distinct hypotheses about the position of the robot then it can be directed to move towards an asymmetry in the environment that will allow one hypothesis to be discounted and the other confirmed. This is known as active rather than passive localisation [73]. In theory a pedestrian localisation system could request that the user move in a certain direction, but this would be highly undesirable in practice.

2.6.2 Map availability

Whilst maps of outdoor environments are widely available, maps of indoor environments are not. This currently prevents robot localisation techniques from being widely applied for the purpose of pedestrian localisation and tracking. Fortunately the availability of such maps is set to increase rapidly over the next few years, with several companies already building such maps for use in navigation and location-based applications. Micello[14] claim to be able to construct a detailed plan of a large shopping centre in only four hours and have already released applications for mobile telephones that allow users to search and navigate within some buildings. NavTeq[15] are also mapping indoor environments such as airports, shopping centres and train stations, using golf buggies and backpacks equipped with a combination of cameras, laser range finders and inertial sensors in order to map quickly and efficiently. They plan to release their first maps of indoor environments in the fourth quarter of 2010 [74]. It is expected that many other mapping companies will also start to map indoor environments, due in part to the perceived demand for location-based services and applications. Established mapping companies such as NavTeq have also been driven to develop new revenue streams

[14] http://www.micello.com
[15] http://www.navteq.com

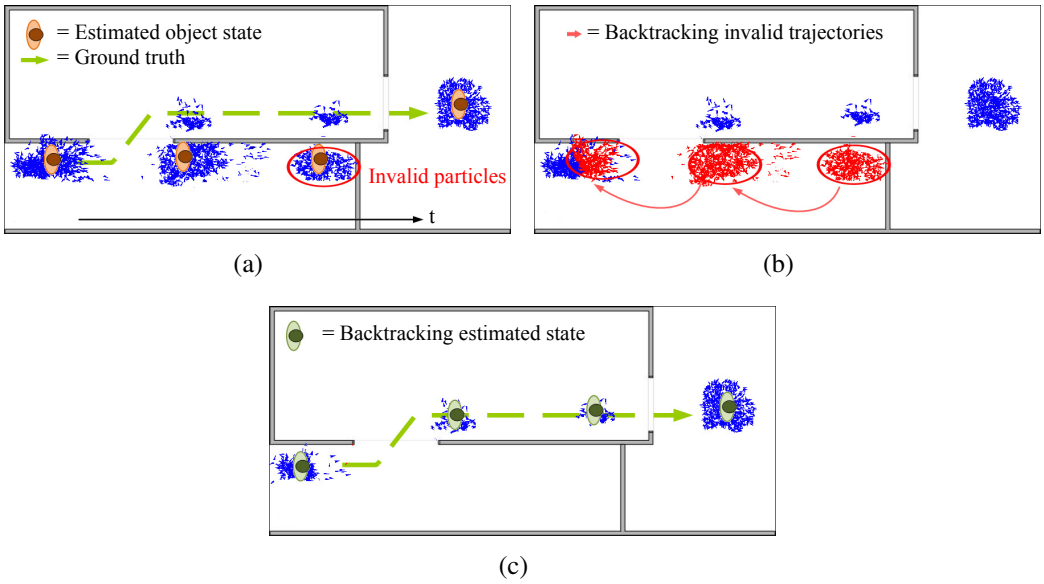


Figure 2.11: The backtracking particle filter. (a) Particles are invalidated as they pass through a wall. (b) The tail states of invalidated (red) particles. (c) The corrected state estimates. Adapted with permission from [77].

since other companies, such as Google, began offering free turn-by-turn satellite navigation applications for mobile telephones. For these reasons it is expected that a large proportion of public buildings in the developed world will be mapped within the next five to ten years.

2.6.3 Concurrent work

Several pedestrian localisation and tracking systems have been developed concurrently to the one presented in this thesis. Krach and Robertson describe a localisation system consisting of two Bayesian filters, an EKF and a particle filter [75]. The EKF extracts relative movement measurements from a foot-mounted IMU, using zero velocity updates as described in Section 2.2.2.2. A second IMU attached to the other foot can be used to improve the accuracy of these measurements [76]. The particle filter fuses such measurements with a map of the environment, a movement model and measurements received from other devices such as an altimeter or digital compass. It is unclear how much of the system has actually been implemented since (at the time of writing) detailed results have yet to be published. In particular their current work does not discuss handling multi-storey buildings or scalability to large environments. At present the only published results are in the form of particle plots, which show the localisation of a pedestrian moving through a single-storey $1000\,\mathrm{m}^2$ office environment [75].

Widyawan et al. describe a backtracking particle filter for combining relative movement measurements from an IMU with a building plan [77, 78]. Each particle in the backtracking filter contains not only its current state, but also a tail of previous states. This allows previous state

estimates to be re-calculated from the tail states of surviving particles each time an update occurs. The re-calculated states are not weighted by particles whose trajectories have been subsequently invalidated, improving the previous estimates as shown in Figure 2.11. Back-tracking was shown to reduce the mean positioning error from $1.553\,\mathrm{m}$ to $1.321\,\mathrm{m}$ during test runs in a $2704\,\mathrm{m}^2$ building for which a detailed map was available [77]. When only the external walls of the building were present in the map the mean error was reduced from $1.892\,\mathrm{m}$ to $1.819\,\mathrm{m}$.

2.7 Conclusions

Whilst satellite positioning systems allow devices to position themselves anywhere in the world when outdoors, they do not work well in indoor environments. This has led to the development of indoor positioning technologies, however the infrastructure requirements of existing indoor location systems make them expensive and time consuming to deploy and maintain. As a result they are not well suited to certain usage scenarios, such as when emergency personnel enter a building to which they have just been called. Temporary deployments (e.g. for a few weeks) of existing indoor positioning systems are impractical due to the high cost of installation. They are also uneconomical for deployment in large, sparsely populated areas, since the cost of such systems is largely dependent on the coverage area rather than the number of tracked users.

In contrast to existing absolute positioning systems, robot localisation algorithms allow robots to determine and track their absolute positions in indoor environments without the need for any fixed infrastructure. Such techniques can in theory be applied to track pedestrians, however they require maps of indoor environments to be available. Until now this has not generally been the case, however increased mapping of such environments by commercial mapping companies is likely to address this problem in the future.

Chapter 3

Inertial navigation

The pedestrian localisation and tracking system developed in this thesis is based on a micro-machined electromechanical systems (MEMS) inertial measurement unit (IMU). This chapter examines these sensors at a low level, develops the theory of inertial navigation, discusses the propagation of errors through inertial navigation systems and finally introduces a filter that is able to correct such systems using external measurements.

MEMS IMUs contain three orthogonal gyroscopes and accelerometers. The gyroscopes measure the angular velocity of the IMU body with respect to inertial space. In other words they measure the IMU's angular velocity with respect to any *inertial* coordinate frame, which is any frame that does not accelerate or rotate with respect to the rest of the universe [18]. The accelerometers measure the specific force acted upon the IMU, also with respect to inertial space. Since the inertial sensors are rigidly attached to the body of the IMU, all measurements are made in the IMU's own frame of reference. This is known as the *body* frame, as shown in

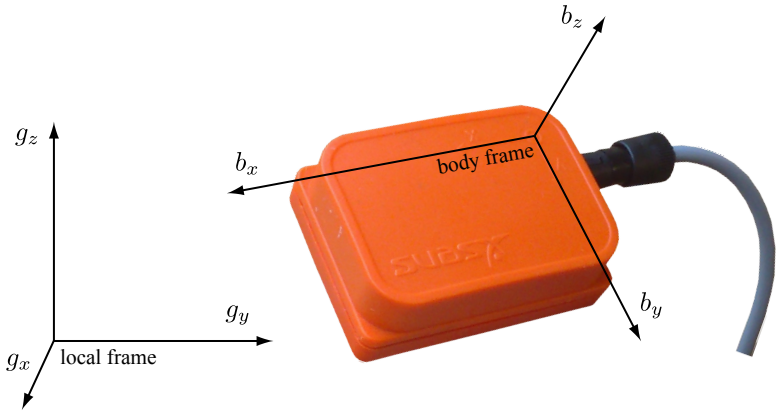


Figure 3.1: An Xsens Mtx together with body and local frames of reference.

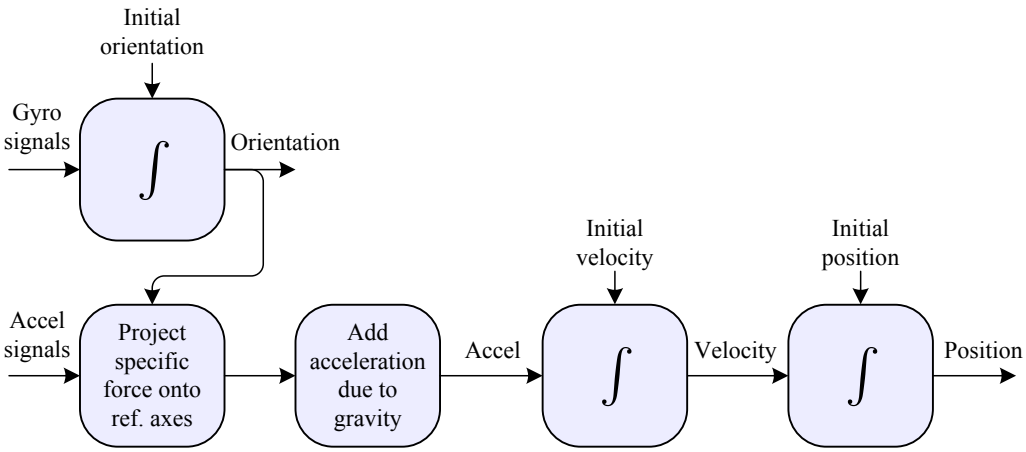


Figure 3.2: The strapdown inertial navigation algorithm.

Figure 3.1. Hence IMUs generate measurements of the form

$$\boldsymbol{\omega}_t^b = (\omega_t^{b_x}, \omega_t^{b_y}, \omega_t^{b_z})^T \tag{3.1}$$

$$\boldsymbol{f}_t^b = (f_t^{b_x}, f_t^{b_y}, f_t^{b_z})^T \tag{3.2}$$

where $\boldsymbol{\omega}_t^b$ and $\boldsymbol{f}_t^b$ are 3-dimensional angular velocity and specific force measurements respectively and t is the time at which the measurements are made. The superscript b indicates that the measurements are in the body frame. Superscripts b_x, b_y and b_z are used to indicate quantities that are measured along the individual axes of the that frame. This superscript notation is adopted throughout the remainder of this chapter.

An inertial navigation system (INS) tracks its orientation, velocity and position in some reference frame that is suitable for the target application. Figure 3.2 gives an overview of the strapdown inertial navigation algorithm that is used. The angular velocity measurements are integrated to track the orientation of the IMU relative to the reference frame. Specific force measurements made by the IMU are projected into this frame using the tracked orientation. Acceleration due to gravity is then added to obtain the acceleration of the IMU in the reference frame. This acceleration is then integrated once to track the velocity of the device and again to track its position [13]. Note that the algorithm must be initialised with an initial state (i.e. orientation, velocity and position).

In the simplest case the reference frame used by an INS is an inertial one, however in practice this is not usually the case. This thesis is concerned with *local* (i.e. within-building) navigation, for which a suitable reference frame is one whose x and y axes are fixed with respect to north and east and whose z axis is locally vertical. Such frames rotate with respect to inertial space due to the rotation of the earth, which must be accounted for by an INS if it is to achieve an accurate navigation solution. Nevertheless there are situations where it is appropriate to ignore such rotation (i.e. assume that a local navigation frame is an inertial frame of reference).

These include situations where the expected gyroscope errors are significantly larger than the error that is introduced as a result of the earth's rotation. It is also appropriate when tracking for short periods of time (e.g. several minutes or less, depending on accuracy requirements) because the earth's rotation will be small during such periods. Another scenario is when the INS is frequently corrected using external measurements that effectively correct for errors that accumulate due to the earth's rotation. Finally, if the INS is used as a purely relative tracking system (i.e. it is not initialised with an absolute position and orientation relative to the earth) then it is not possible to compensate for the earth's rotation, since knowledge of the latitude and absolute orientation of the device is required (see [13] for details).

The tracking system developed in this thesis uses inertial navigation as a purely relative tracking technique in order to measure changes in a pedestrian's position and orientation. The absolute position and heading of the user are (at least initially) unknown and hence it is not possible to compensate for the earth's rotation. Furthermore the errors introduced by the Xsens Mtx IMU that is used are significantly larger than those introduced by ignoring the earth's rotation, as will be shown in Section 3.3.1. As a result local navigation frames are approximated as inertial frames of reference in this thesis. Quantities measured in a local frame are indicated by a superscript g. All frames of reference are assumed to be right-handed[1], as shown in Figure 3.1.

The remainder of this chapter is arranged as follows. Section 3.1 describes error sources that perturb measurements from the individual sensors contained within a MEMS IMU. Section 3.2 develops the strapdown inertial navigation algorithm outlined in Figure 3.2. The propagation of errors through this algorithm is discussed in Section 3.3. Finally, a Kalman filter that can correct an INS using external measurements is developed and tested in Section 3.4.

3.1 Sources of error in MEMS inertial sensors

Various types of error can perturb measurements generated by the individual gyroscopes and accelerometers that are contained within an IMU. The main types of error that affect measurements made by MEMS sensors are listed below. Since an inertial navigation system must integrate signals from such sensors in order to track the state of an IMU, the effect of integrating signals containing each type of error is also considered. Note that in this section these integrations are considered only for a single axis of the device. A more general discussion of errors and how they propagate through the complete inertial navigation algorithm can be found in Section 3.3.

[1] As opposed to left-handed, in which the x and y axes (or indeed any other pair of axes) are exchanged.

3.1.1 Gyroscope error sources

3.1.1.1 Constant bias

The bias of a gyroscope is the average output from the gyroscope when it is not undergoing any rotation (i.e. the difference between the output and the true value), usually specified in $^\circ/\mathrm{h}$. A constant bias error of ϵ, when integrated, causes an angular error that grows linearly with time

$$\theta(t) = \epsilon \cdot t. \tag{3.3}$$

The constant bias of a gyroscope can be estimated by taking a long term average of its output when it is not undergoing rotation. This estimate can then be subtracted from any subsequent measurements. Note however that the presence of other errors means that it is impossible to determine the bias of a gyroscope exactly.

3.1.1.2 Random noise / angle random walk

MEMS gyroscopes are perturbed by random noise from a number of sources, including themo-mechanical and electrical noise. The spectrum of such noise is approximately white for frequencies below $1\,\mathrm{Hz}$, meaning that the standard deviation of the average angular velocity noise is inversely proportional to the square root of the averaging time [18]. Thus gyroscope sensor noise is usually specified in root power spectral density (PSD) form with units $^\circ/\mathrm{h}/\sqrt{\mathrm{Hz}}$, $^\circ/\sqrt{\mathrm{h}}$ or $^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}}$ (which is equivalent to $^\circ/\sqrt{\mathrm{s}}$) for less accurate sensors. The relationship between root PSD n (expressed in each of the units above) and the standard deviation σ of the individual measurements is given by

$$\sigma\,(^\circ/s) = \frac{n\,(^\circ/\mathrm{h}/\sqrt{\mathrm{Hz}})}{60 \cdot 60 \cdot \sqrt{\delta t\,(\mathrm{s})}} \tag{3.4}$$

$$= \frac{n\,(^\circ/\sqrt{\mathrm{h}})}{60 \cdot \sqrt{\delta t\,(\mathrm{s})}} \tag{3.5}$$

$$= \frac{n\,(^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}})}{\sqrt{\delta t\,(\mathrm{s})}} \tag{3.6}$$

where δt is the period between successive samples (e.g. $0.01\,\mathrm{s}$ if the sampling rate is $100\,\mathrm{Hz}$).

The result of integrating the random noise that perturbs MEMS gyroscopes is a random walk in angle. The mean of the resulting random walk is zero, but its standard deviation grows proportional to $\sqrt{t}$. For example if the root PSD is $0.2^\circ/\sqrt{\mathrm{h}}$ then the standard deviation of the angle random walk will be 0.2° after one hour, $\sqrt{2} \cdot 0.2 = 0.28^\circ$ after two hours and so on.

3.1.1.3 Calibration errors

Calibration errors include errors in the scale factors, alignments, and linearities of the gyroscopes. Such errors tend to produce systematic errors that are only observed whilst the device is turning and cause the accumulation of errors in the integrated signal that are proportional to the rate and duration of the motions [42]. It is usually possible to measure and correct such calibration errors. The Xsens Mtx applies its own internal corrections to reduce the effect of calibration errors.

3.1.1.4 Temperature effects

Temperature fluctuations due to changes in the environment and sensor self-heating can induce small fluctuations in the sensor bias. Any residual bias introduced due to a change in temperature will cause an error in orientation which grows linearly with time. The relationship between bias and temperature is often highly non-linear for MEMS sensors. Like many other IMUs the Xsens Mtx contains a temperature sensor that it uses to minimise bias fluctuations induced by changes in temperature.

3.1.1.5 Bias instability

The bias of a MEMS gyroscope tends to wander slowly over time even when the temperature of the device is kept constant. Such fluctuations are often modelled as a random walk [42]. They are quantified by a bias instability measurement that defines how the bias may change over a specified period of time, typically around 100 seconds, in fixed conditions. Most manufacturers provide a 1σ value in $^\circ/\mathrm{h}$, or $^\circ/\mathrm{s}$ for less accurate devices. If B_t is the known bias at time t, then a 1σ bias instability of $\mathrm{BS}_\sigma = 0.01^\circ/\mathrm{h}$ over 100 seconds means that the bias at time $(t + 100)$ seconds is a random variable with expected value B_t and standard deviation $0.01^\circ/\mathrm{h}$. Under the random walk model this standard deviation grows proportional to the square root of time. For this reason bias instability is occasionally specified as a bias random walk measurement

$$\text{BRW}\,(^\circ/\sqrt{\mathrm{h}}) = \frac{\mathrm{BS}_\sigma\,(^\circ/\mathrm{h})}{\sqrt{t\,(\mathrm{h})}} \tag{3.7}$$

where t is the timespan for which BS_σ is defined.

Under the assumption that bias instabilities can be modelled as a random walk, the resulting error in the integrated signal is a second-order random walk[2] in angle. Note however that this model is unrealistic for running times that are much larger than the timespan over which BS_σ is defined, since uncertainty in the bias grows without bound in the random walk model

[2]The integral of a first-order random walk.

Error type	Description	Result of single integration
Bias	A constant bias offset	A linearly growing error in angle
Random noise	Random noise from a variety of sources (usually specified in terms of its root PSD)	An angle random walk whose standard deviation grows proportional to the square root of time
Calibration	Deterministic errors in scale factors, alignments and gyro linearities	Orientation drift proportional to the rate and duration of motion
Temperature effects	Temperature dependent residual bias	Any residual bias is integrated into the orientation, causing an orientation error which grows linearly with time
Bias instability	Bias fluctuations, usually modelled as a bias random walk	A second-order random walk

Table 3.1: A summary of MEMS gyroscope error sources.

whereas in reality the bias will always remain within some range around zero. More accurate (but more processor intensive) approaches include modelling bias instability using Markov processes and representing sensor biases using second-order and third-order autoregressive models [18].

A summary of the different error sources described above is shown in Table 3.1.

3.1.2 Accelerometer error sources

The main types of error that perturb measurements from MEMS accelerometers are the same as those described above for MEMS gyroscopes. The primary difference is that errors in acceleration measurements are integrated *twice* to track the position of an IMU, whereas angular velocity measurements are only integrated once to track the estimated orientation. As a result bias errors cause errors in position that grow proportional to t^2, white noise causes a second-order random walk in position and so on. A summary of the different error sources can be found in Table 3.2. More detailed information can be found in [79].

3.2 Strapdown inertial navigation

This section develops the strapdown inertial navigation algorithm that is illustrated by Figure 3.2. The equations necessary to track the state of an IMU over time are first derived for the theoretical case in which continuous angular velocity and acceleration signals are used. The real-life case in which an IMU provides samples of these signals at discrete points in time is discussed in Section 3.2.2. The derivations in both sections are based on those in [13]. Note

Error type	Description	Result of double integration
Bias	A constant bias offset	A position error that grows proportional to t^2
Random noise	Random noise from a variety of sources (usually specified in terms of its root PSD)	A second-order random walk with a zero mean and a standard deviation that grows proportional to $t^{3/2}$.
Calibration	Deterministic errors in scale factors, alignments and accelerometer linearities	Position drift proportional to the squared rate and duration of acceleration
Temperature effects	Temperature dependent residual bias	Any residual bias causes an error in position that grows quadratically with time
Bias instability	Bias fluctuations, usually modelled as a bias random walk	A third-order random walk in position

Table 3.2: A summary of MEMS linear accelerometer error sources.

that the local navigation frame is approximated as an inertial frame of reference in this section. Hence compensation terms due to the rotation of the earth are omitted.

3.2.1 Theory

The orientation, or *attitude*, of an IMU relative to a local reference frame can be tracked by integrating its angular velocity

$$\boldsymbol{\omega}^b(t) = (\omega^{b_x}(t), \omega^{b_y}(t), \omega^{b_z}(t))^T . \tag{3.8}$$

The orientation of the device can be specified using several different representations, including Euler angles, quaternions and direction cosines. The direction cosines representation is used in the derivations below. Equivalent derivations using both Euler angles and quaternions can be found in [13].

In the direction cosines representation the orientation of the IMU's body frame relative to the local frame is specified by a 3×3 rotation matrix $\boldsymbol{C}$, in which each column is a unit vector along one of the body axes specified in terms of the local axes. A vector quantity $\boldsymbol{v}^b$ defined in the body frame is equivalent to the vector

$$\boldsymbol{v}^g = \boldsymbol{C}\boldsymbol{v}^b \tag{3.9}$$

defined in the local frame. The inverse transformation is given by

$$\boldsymbol{v}^b = \boldsymbol{C}^T\boldsymbol{v}^g \tag{3.10}$$

since the inverse of a rotation matrix is equivalent to its transpose.

If the orientation of the IMU at time t is given by $\boldsymbol{C}(t)$ then the rate of change of $\boldsymbol{C}$ at t is given by

$$\dot{\boldsymbol{C}}(t) = \lim_{\delta t \to 0} \frac{\boldsymbol{C}(t+\delta t) - \boldsymbol{C}(t)}{\delta t} \tag{3.11}$$

in which $\boldsymbol{C}(t+\delta t)$ can be written as the product of two matrices

$$\boldsymbol{C}(t+\delta t) = \boldsymbol{C}(t)\boldsymbol{A}(t) \tag{3.12}$$

where $\boldsymbol{A}(t)$ is the rotation matrix that relates the body frame at time t to the body frame at time $(t+\delta t)$ [13]. If $\delta\phi$, $\delta\theta$ and $\delta\psi$ are the small clockwise rotations through which the body frame has rotated between times t and $(t+\delta t)$ about its x, y and z axes respectively, then using the small angle approximation (see Appendix B.1) $\boldsymbol{A}(t)$ can be written as

$$\boldsymbol{A}(t) = \boldsymbol{I} + \boldsymbol{\delta\Psi} \tag{3.13}$$

where

$$\boldsymbol{\delta\Psi} = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{pmatrix}. \tag{3.14}$$

Hence by substitution

$$\begin{aligned} \dot{\boldsymbol{C}}(t) &= \lim_{\delta t \to 0} \frac{\boldsymbol{C}(t+\delta t) - \boldsymbol{C}(t)}{\delta t} && (3.15)\\ &= \lim_{\delta t \to 0} \frac{\boldsymbol{C}(t)\boldsymbol{A}(t) - \boldsymbol{C}(t)}{\delta t} && (3.16)\\ &= \lim_{\delta t \to 0} \frac{\boldsymbol{C}(t)(\boldsymbol{I} + \boldsymbol{\delta\Psi}) - \boldsymbol{C}(t)}{\delta t} && (3.17)\\ &= \boldsymbol{C}(t) \lim_{\delta t \to 0} \frac{\boldsymbol{\delta\Psi}}{\delta t}. && (3.18) \end{aligned}$$

In the limit $\delta t \to 0$ the small angle approximation is valid, and

$$\lim_{\delta t \to 0} \frac{\boldsymbol{\delta\Psi}}{\delta t} = \boldsymbol{\Omega}(t) \tag{3.19}$$

where

$$\boldsymbol{\Omega}(t) = \begin{pmatrix} 0 & -\omega^{b_z}(t) & \omega^{b_y}(t) \\ \omega^{b_z}(t) & 0 & -\omega^{b_x}(t) \\ -\omega^{b_y}(t) & \omega^{b_x}(t) & 0 \end{pmatrix} \tag{3.20}$$

is the skew symmetric form of the angular velocity $\boldsymbol{\omega}^b(t)$. Hence in order to track the orienta-

tion it is necessary to solve the differential equation

$$\dot{\boldsymbol{C}}(t) = \boldsymbol{C}(t)\boldsymbol{\Omega}(t) \tag{3.21}$$

which has the solution

$$\boldsymbol{C}(t) = \boldsymbol{C}(0) \cdot \exp\left(\int_0^t \boldsymbol{\Omega}(t)\,\mathrm{dt}\right) \tag{3.22}$$

where $\boldsymbol{C}(0)$ is the initial orientation of the device [13].

To track the position of an IMU the specific force

$$\boldsymbol{f}^b(t) = (f^{b_x}(t), f^{b_y}(t), f^{b_z}(t))^T \tag{3.23}$$

is first projected into the local frame of reference

$$\boldsymbol{f}^g(t) = \boldsymbol{C}(t)\boldsymbol{f}^b(t)\,. \tag{3.24}$$

Acceleration due to gravity is then added to obtain the acceleration of the IMU, which is integrated once to obtain velocity and again to obtain displacement

$$\boldsymbol{a}^g(t) = \boldsymbol{f}^g(t) + \boldsymbol{g}^g \tag{3.25}$$

$$\boldsymbol{v}^g(t) = \boldsymbol{v}^g(0) + \int_0^t \boldsymbol{a}^g(t)\,\mathrm{dt} \tag{3.26}$$

$$\boldsymbol{s}^g(t) = \boldsymbol{s}^g(0) + \int_0^t \boldsymbol{v}^g(t)\,\mathrm{dt} \tag{3.27}$$

where $\boldsymbol{v}^g(0)$ is the initial velocity of the device, $\boldsymbol{s}^g(0)$ is the initial displacement and $\boldsymbol{g}^g$ is acceleration due to gravity specified in the local frame of reference.

3.2.2 Implementation

Since an IMU only provides fixed-rate samples of the continuous signals $\boldsymbol{\omega}^b(t)$ and $\boldsymbol{a}^b(t)$, integration schemes must be used to apply Equations 3.22, 3.26 and 3.27 in practice [13]. The choice of scheme is application dependent. For short timespan and low accuracy applications a low order integration scheme such as the rectangular rule is sufficient. For more demanding applications a third or fourth-order scheme may be more appropriate. This section develops an INS implementation that uses the rectangular rule, since it is computationally simple and is adequate for use given the current quality of MEMS inertial sensors.

Let the period between successive samples be δt. For a single sampling period $(t - \delta t, t)$ the

solution to Equation 3.21 can be written as

$$\boldsymbol{C}_t = \boldsymbol{C}_{t-\delta t} \cdot \exp\left(\int_{t-\delta t}^{t} \boldsymbol{\Omega}(t)\,\mathrm{dt}\right). \tag{3.28}$$

The rectangular rule gives

$$\int_{t-\delta t}^{t} \boldsymbol{\Omega}(t)\,\mathrm{dt} = \boldsymbol{B}_t \tag{3.29}$$

where

$$\boldsymbol{B}_t = \begin{pmatrix} 0 & -\omega_t^{b_z}\delta t & \omega_t^{b_y}\delta t \\ \omega_t^{b_z}\delta t & 0 & -\omega_t^{b_x}\delta t \\ -\omega_t^{b_y}\delta t & \omega_t^{b_x}\delta t & 0 \end{pmatrix} \tag{3.30}$$

in which $\boldsymbol{\omega}_t^b = (\omega_t^{b_x}, \omega_t^{b_y}, \omega_t^{b_z})^T$ is the angular velocity measurement corresponding to the update period. Letting $\sigma = |\boldsymbol{\omega}_t^b \delta t|$, substituting Equation 3.29 into Equation 3.28 and finally performing the Taylor expansion of the exponential term gives

$$\boldsymbol{C}_t = \boldsymbol{C}_{t-\delta t}\left(\boldsymbol{I} + \boldsymbol{B}_t + \frac{\boldsymbol{B}_t^2}{2!} + \frac{\boldsymbol{B}_t^3}{3!} + \frac{\boldsymbol{B}_t^4}{4!} + ...\right) \tag{3.31}$$

$$= \boldsymbol{C}_{t-\delta t}\left(\boldsymbol{I} + \boldsymbol{B}_t + \frac{\boldsymbol{B}_t^2}{2!} - \frac{\sigma^2\boldsymbol{B}_t}{3!} - \frac{\sigma^2\boldsymbol{B}_t^2}{4!} + ...\right) \tag{3.32}$$

$$= \boldsymbol{C}_{t-\delta t}\left(\boldsymbol{I} + \left(1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!}...\right)\boldsymbol{B}_t + \left(\frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!}...\right)\boldsymbol{B}_t^2\right) \tag{3.33}$$

$$= \boldsymbol{C}_{t-\delta t}\left(\boldsymbol{I} + \frac{\sin\sigma}{\sigma}\boldsymbol{B}_t + \frac{1-\cos\sigma}{\sigma^2}\boldsymbol{B}_t^2\right). \tag{3.34}$$

This is the orientation update equation that is applied by an INS each time a new measurement is received from the IMU [13].

To update the tracked position of the IMU the specific force measurement $\boldsymbol{f}_t^b$ is first projected into the local frame of reference

$$\boldsymbol{f}_t^g = \boldsymbol{C}_t \boldsymbol{f}_t^b. \tag{3.35}$$

The velocity and displacement components of the state are then updated. Using the rectangular rule the update Equations 3.25, 3.26 and 3.27 become

$$\boldsymbol{a}_t^g = \boldsymbol{f}_t^g + \boldsymbol{g}^g \tag{3.36}$$

$$\boldsymbol{v}_t^g = \boldsymbol{v}_{t-\delta t}^g + \delta t \cdot \boldsymbol{a}_t^g \tag{3.37}$$

$$\boldsymbol{s}_t^g = \boldsymbol{s}_{t-\delta t}^g + \delta t \cdot \boldsymbol{v}_t^g. \tag{3.38}$$

Equations 3.34, 3.35, 3.36, 3.37 and 3.38 are all that an INS must implement to update the estimated orientation, velocity and displacement of the tracked IMU each time a new measurement is received. Note however that the INS must still be provided with the initial state

$(\boldsymbol{C}_0, \boldsymbol{v}_0^g, \boldsymbol{s}_0^g)$ of the device.

3.3 Error propagation in inertial navigation systems

When using an INS the most important property that must be considered is the rate at which drift errors accumulate in the calculated position. The most obvious cause of drift is the propagation of errors perturbing the acceleration measurements through the double integration, as summarised in Table 3.2. For example a constant bias error affecting a single accelerometer will cause drift in position that grows proportional to t^2. Although this is the most obvious cause of drift, it is not the most severe except during the early stages of inertial navigation. Over longer time periods errors in orientation that arise due to errors perturbing the angular velocity measurements account for a greater proportion of drift (with the crossover point dependent on the relative quality of the accelerometers and gyroscopes used). This is due to the use of the calculated orientation $\boldsymbol{C}_t$ to project specific force measurements into the global frame of reference. Suppose that errors in the angular velocity measurements cause a small tilt error ϵ in $\boldsymbol{C}_t$. After acceleration due to gravity is added in the local frame, $\boldsymbol{a}_t^g$ will contain a residual bias of magnitude $g \cdot \sin(\epsilon)$ in the (x, y) plane and $g \cdot (1 - \cos(\epsilon))$ along the z axis. These errors will then be integrated into both the velocity and displacement components of the INS state. Note that if ϵ is small then $\cos(\epsilon) \approx 1$ and $\sin(\epsilon) \approx \epsilon$. Hence the majority of drift caused by small tilt errors occurs in the (x, y) plane.

Note that even a small error in orientation causes a rapidly growing error in the calculated position. For example consider a constant tilt error of just $0.05°$. This error will cause a component of acceleration due to gravity with magnitude $0.0086\,\mathrm{m/s^2}$ to be projected onto the horizontal axes. This residual bias will in turn cause drift in the (x, y) plane that grows quadratically to $7.7\,\mathrm{m}$ after only 30 seconds. In practice such tilt errors are often introduced by small bias errors in the gyroscope signals. In such cases the tilt errors are themselves growing linearly with time, meaning that the resulting drift in position will grow proportional to t^3. Note also that if the local frame is approximated as an inertial frame of reference (as is the case in this thesis), then the earth's rotation will also introduce an error in orientation that grows linearly with time.

3.3.1 Xsens Mtx example

To demonstrate the relative importance of different error sources and their propagation through the INS equations, the performance of an INS based on an Xsens Mtx IMU was analysed for the stationary case. The INS navigation processor, which applied the equations given in Section 3.2.2 to update the estimated state, was implemented within an event-based framework

written in the Java programming language. A Java interface to the Xsens Mtx was also written within this framework to allow measurement data from the device to be processed in real time. Event logging and replay facilities allowed event streams to be recorded and replayed at a later date.

To test the performance of the INS it was applied to 100 streams of data, each obtained from a stationary Mtx device with a sampling frequency of $100\,\mathrm{Hz}$. Each stream was 135 seconds in length with the first $15\,\mathrm{s}$ (from $t = -15\,\mathrm{s}$ to $t = 0\,\mathrm{s}$) used for initialisation in each case. The first 10 seconds of the initialisation process was used to calculate the average specific force vector measured by the IMU in the body frame. The INS attitude was then initialised at $t = -5\,\mathrm{s}$ such that this vector was aligned[3] with the z-axis of the local frame of reference. The heading component of the orientation was initialised to an arbitrary value. Note that this levelling process is not exact, since the average specific force vector will be affected by bias errors perturbing the accelerometer signals.

If the INS were run after levelling as described above, drift arising due to the accelerometer biases would accumulate in the opposite direction to drift caused by the initial tilt error (which is introduced by the levelling process as a result of the same accelerometer biases). Hence the two would tend to cancel out, resulting in an overly optimistic estimate of true performance. To address this problem the IMU was rotated by 90° between $t = -5\,\mathrm{s}$ and $t = 0\,\mathrm{s}$ of each run. This causes the two error sources to act at 90° to one another rather than in opposite (cancelling) directions. At $t = 0\,\mathrm{s}$ of each run the global displacement and velocity were initialised to zero

$$\boldsymbol{s}_0^g = \boldsymbol{v}_0^g = (0, 0, 0)^T . \tag{3.39}$$

The INS was then used to track the position of the IMU whilst it was kept stationary from $t = 0\,\mathrm{s}$ until $t = 120\,\mathrm{s}$.

Since the IMU was stationary during each run (excluding the initialisation phase), the total drift in position at time t during a particular run of the INS is given by $|\boldsymbol{s}_t^g|$. Figure 3.3 shows how this drift (averaged over all 100 runs) increased over time. After $60\,\mathrm{s}$ the average drift was $170\,\mathrm{m}$ and by $120\,\mathrm{s}$ it had grown to $1052\,\mathrm{m}$. The average drift along the global z-axis at $t = 120\,\mathrm{s}$ was $|s_t^{g_z}| = 168.7\,\mathrm{m}$, compared to a much larger average drift of $1031\,\mathrm{m}$ in the (x, y) plane. This is typical of drift caused by primarily by the propagation of small tilt errors, as described in Section 3.3. Such tilt errors arise due to both gyroscope errors and the rotation of the earth, which is not compensated for by the INS.

To analyse the underlying causes of the accumulating drift, each of the streams of data were reprocessed by the INS whilst selectively removing error sources. This was possible because it was known that the IMU remained stationary during each run. Accelerometer errors were

[3] An algorithm for deriving a rotation matrix that aligns two vectors can be found in Appendix B.3.

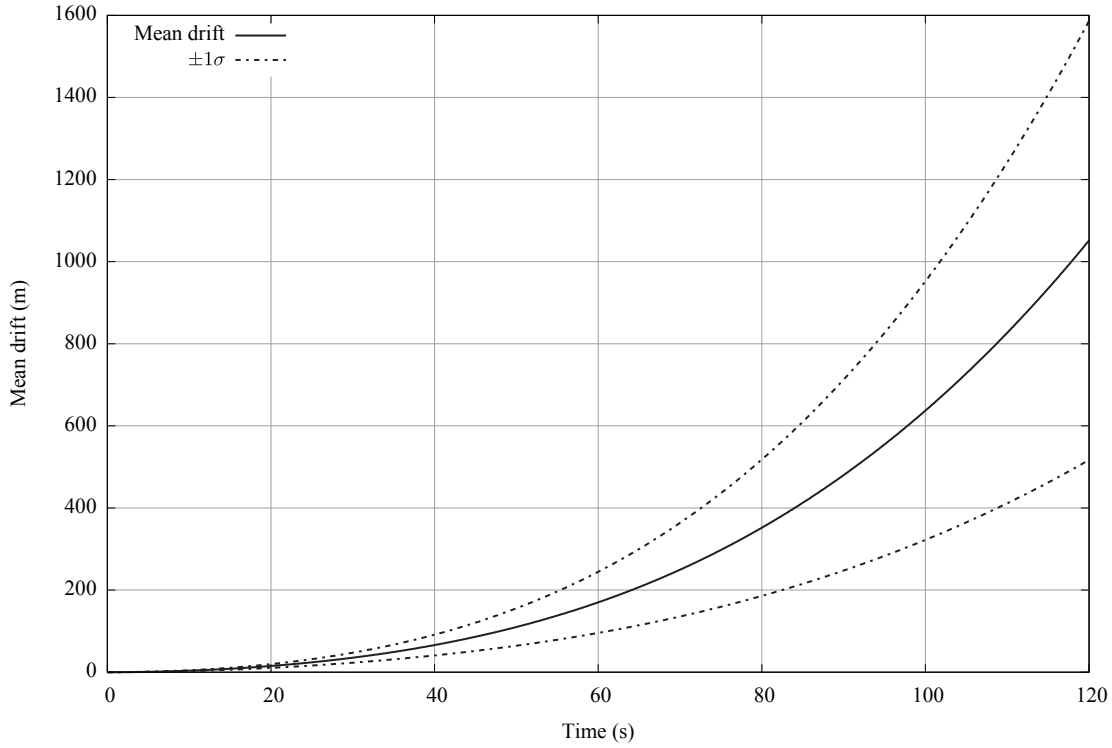


Figure 3.3: The mean drift incurred by an INS during 100 test runs using an Xsens Mtx IMU. Each run was of length 120 s.

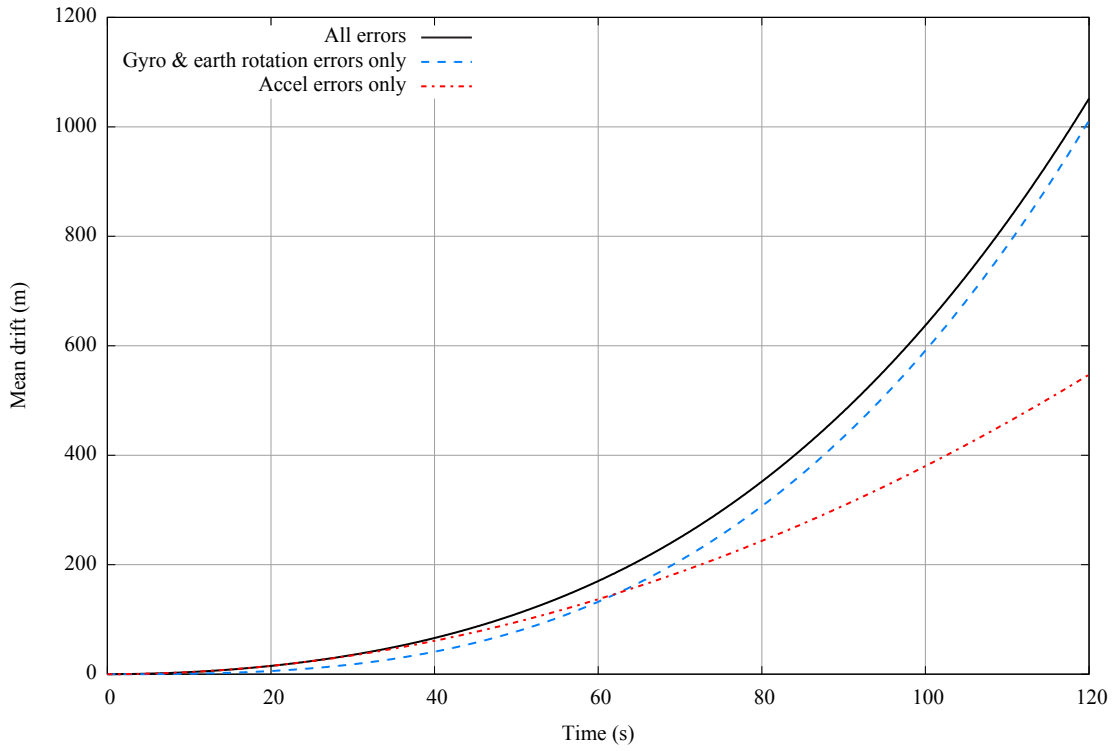


Figure 3.4: The mean drifts incurred when error sources were selectively removed.

removed by setting every specific force measurement from $t = 0\,\mathrm{s}$ to $t = 120\,\mathrm{s}$ to

$$\boldsymbol{f}_t^b = -\boldsymbol{C}_0^T \boldsymbol{g}^g \tag{3.40}$$

where $\boldsymbol{C}_0^T$ is the orientation of the IMU after initialisation and $\boldsymbol{g}^g = (0, 0, -9.807)^T$ is acceleration due to gravity. Errors arising from the gyroscopes (and as a result of the earth's rotation) were removed by setting every angular velocity measurement from $t = 0\,\mathrm{s}$ to $t = 120\,\mathrm{s}$ to

$$\boldsymbol{\omega}_t^b = (0, 0, 0)^T . \tag{3.41}$$

Figure 3.4 shows the average drift curves obtained when the error sources were selectively removed. Note that the accelerometers are the primary cause of drift up until $t = 62\,\mathrm{s}$, after which the drift arising due to the combined effect of gyroscope errors and the earth's rotation become dominant. The curve corresponding to errors perturbing the accelerometers grows almost exactly in proportion to t^2, which is the rate expected due to the propagation of accelerometer bias errors through the INS. The curve corresponding to errors perturbing the gyroscopes and the earth's rotation grows almost exactly in proportion to t^3, which is the rate expected due to the propagation of gyroscope bias errors through the INS. At $t = 120\,\mathrm{s}$ the average drift caused by the accelerometers was $547\,\mathrm{m}$, whilst the average drift caused by the gyroscopes and the earth's rotation was $1012\,\mathrm{m}$.

In the case of drift caused by errors in orientation, it is possible to quantify how much of the drift is caused by the earth's rotation as opposed to the gyroscopes. This can be achieved by initialising the INS at $t = 0\,\mathrm{s}$ with

$$\boldsymbol{v}_0^g = (0, 0, 0)^T \tag{3.42}$$

$$\boldsymbol{s}_0^g = (0, 0, 0)^T \tag{3.43}$$

$$\boldsymbol{C}_0 = \boldsymbol{I} \tag{3.44}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix. The INS is then run for $120\,\mathrm{s}$ during which every measurement is given by

$$\boldsymbol{f}_t^b = -\boldsymbol{g}^g \tag{3.45}$$

$$\boldsymbol{\omega}_t^b = \boldsymbol{C}_0^T \cdot (\Omega \cos L, 0, -\Omega \sin L)^T \tag{3.46}$$

in which $L = 52.2^\circ$ is the latitude of the location where the tests were performed (Cambridge, UK) and $\Omega = 7.2921 \times 10^{-5}\,\mathrm{rad/s}$ is the earth's rate of rotation [13]. This simulates a stationary INS in which the only error source is a growing error in orientation caused by the rotation of the earth. After $120\,\mathrm{s}$ the drift in position calculated by the simulation is $125\,\mathrm{m}$. Hence the vast majority of the $1012\,\mathrm{m}$ drift caused by orientation errors in the example above

can be attributed to errors perturbing the gyroscope signals rather than to the rotation of the earth. This means that even if the INS compensated for the earth's rotation, the drift in position would only be reduced by a small fraction of its total value.

There are two main conclusions that can be drawn from the results above. The first is that errors perturbing the accelerometer signals are the primary source of drift in the INS for just over the first minute of operation, after which errors perturbing the gyroscope signals become dominant. The second and most important conclusion is that MEMS IMUs are not yet sufficiently accurate to track the position of a device or person to sub-metre accuracy for extended periods of time (any period greater than $5\,\mathrm{s}$ for the tested IMU) using inertial navigation alone. Although MEMS IMUs are improving rapidly, the t^3 long term error-growth means that even major improvements in the quality of such sensors will yield only modest increases in the durations for which tracking can be performed before error tolerances are exceeded. Finally, note that drift in position will be significantly worse when tracking moving objects rather than stationary ones. This is largely due to errors in the tracked heading, which only propagate to errors in position when the IMU is moving. In addition gyroscope non-linearities and scale factor errors are only observed when the IMU is rotating. These will cause additional errors to accumulate in the orientation, which will in turn result in additional drift in the tracked position.

3.4 INS error correction

One technique to allow a MEMS-based INS to track the position of an object or person for an extended period of time is to regularly correct its state using external measurements. For example positions from an absolute positioning system can be used to periodically correct errors that would otherwise accumulate in the calculated position. The naïve way to apply such measurements is to directly correct the INS state. For example an absolute position measurement can be applied by simply setting the displacement component of the state to equal the displacement reported by the measurement. Although the simplicity of this approach is attractive, it suffers from two major problems:

- External measurements that can be used to correct the system are usually inexact. The naïve approach to INS correction does not account for uncertainty in external measurements.
- The only part of the INS state to be corrected by a measurement is the part that it directly observes. For example an absolute position measurement is only used to correct the position component of the state. An INS that is periodically corrected in this way will still accumulate errors in the orientation and velocity components, which will not

be corrected. It should be possible to do much better than this because the propagation of errors through the INS equations causes correlations between errors in different components of the state. In particular tilt errors in the orientation are strongly correlated to errors in both velocity and displacement. By keeping track of such correlations it should be possible to use measurements that observe one component of the state to correct correlated errors in other components that are not directly observed.

Both these problems can be addressed by using a Bayesian filter to track the state of an INS and to incorporate external measurements. Recall from Section 2.3 that a Bayesian filter takes the uncertainty of measurements into account. Bayesian filters are also able to track correlations between errors that accumulate in different parts of the state. This is possible because a Bayesian filter must be provided with a propagation model that describes how the state of the system evolves over time, from which such correlations can be deduced.

This section describes a simple Kalman filter that is able to correct an INS using external measurements. The Kalman filter is well suited to this task when compared to the other types of Bayesian filter described in Section 2.3. Unlike both grid-based filters and particle filters they are sufficiently computationally efficient to support the high update rates required without the need for expensive hardware. A uni-modal representation of belief is sufficient and hence the added complexity of Gaussian mixture models is unnecessary.

Rather than tracking the total state (i.e. orientation, velocity and position) of the IMU directly, the Kalman filter described below maintains a probability distribution representing the error in the state calculated by a regular INS. Each time a new measurement $(\boldsymbol{\omega}_t^b, \boldsymbol{f}_t^b)$ is received from the IMU it is used to update the INS state as normal. The Kalman filter's belief of the *error-state* is also propagated forward to the current time t. When an external measurement is received it is used to update the estimated error-state during the filter's correction step. This estimate is then immediately transferred to the underlying INS before being reset to zero (this is known as *closed-loop* correction). Using an error-state filter together with a separate INS is mathematically equivalent to an extended Kalman filter (EKF) that tracks the total state directly (see Appendix C). It is a convenient approach because it allows existing INS software components such as those developed and tested in Section 3.3.1 to be re-used with minimal effort, rather than having to replicate the functionality within an equivalent total-state filter.

The remainder of this section describes the error-state Kalman filter and its interaction with the underlying INS in more detail. Section 3.4.1 defines the error-state and its relationship with the inertial navigation solution estimated by the INS. The filter's propagation step, which is executed whenever a new measurement is received from the IMU, is outlined in Section 3.4.2. The correction step that is executed whenever an external measurement is received is described in Section 3.4.3. Finally, Section 3.4.4 describes how the estimated error-state is transferred from the filter to the underlying INS following each correction step.

3.4.1 Error-state definition

The true state of an IMU at time t can be defined in terms of its orientation, displacement and velocity

$$(\boldsymbol{C}_t, \boldsymbol{v}_t, \boldsymbol{s}_t)\,. \tag{3.47}$$

The rotation matrix $\boldsymbol{C}_t$ defines the orientation of the IMU relative to the local frame of reference, as described in Section 3.2. The displacement and velocity are both defined in the local frame. Superscripts indicating the frame of reference in which each quantity is specified are intentionally omitted in this section in order to avoid notational clutter.

It is not possible for an INS to calculate the true state of its IMU (if it could then it would be able to accurately track its position indefinitely without the need for external correction). The inertial navigation solution calculated by the INS diverges from the true state of the IMU due to the propagation of errors through the inertial navigation equations, as described in Section 3.3. In this section the solution calculated by the INS is denoted

$$(\hat{\boldsymbol{C}}_t, \hat{\boldsymbol{v}}_t, \hat{\boldsymbol{s}}_t) \tag{3.48}$$

where hat symbols are used to distinguish INS estimates from the true state of the IMU. The error-state is the difference between the true state and the solution calculated by the INS. It is defined in column vector form as

$$\delta\boldsymbol{x}_t = (\delta\boldsymbol{\phi}_t^T, \delta\boldsymbol{v}_t^T, \delta\boldsymbol{s}_t^T)^T \tag{3.49}$$

where $\delta\boldsymbol{v}_t$ and $\delta\boldsymbol{s}_t$ are errors in velocity and displacement respectively. The error in orientation is expressed such that a clockwise rotation of magnitude $|\delta\boldsymbol{\phi}_t|$ about $\delta\boldsymbol{\phi}_t$ in the local frame of reference will correct the error. Formally, the error-state is defined such that

$$\boldsymbol{C}_t = \boldsymbol{\Phi}_t\hat{\boldsymbol{C}}_t \tag{3.50}$$

$$\boldsymbol{v}_t = \hat{\boldsymbol{v}}_t + \delta\boldsymbol{v}_t \tag{3.51}$$

$$\boldsymbol{s}_t = \hat{\boldsymbol{s}}_t + \delta\boldsymbol{s}_t \tag{3.52}$$

where $\boldsymbol{\Phi}_t$ is the matrix form of the rotation defined by $\delta\boldsymbol{\phi}_t = (\delta\phi_t^{g_x}, \delta\phi_t^{g_y}, \delta\phi_t^{g_z})$. If the rotation is small (as will be the case if the INS solution is frequently corrected using external measurements) then $\boldsymbol{\Phi}_t$ can be approximated as

$$\boldsymbol{\Phi}_t \approx \begin{pmatrix} 1 & -\delta\phi_t^{g_z} & \delta\phi_t^{g_y} \\ \delta\phi_t^{g_z} & 1 & -\delta\phi_t^{g_x} \\ -\delta\phi_t^{g_y} & \delta\phi_t^{g_x} & 1 \end{pmatrix} \tag{3.53}$$

in which small angle approximations are used. An exact form suitable for larger rotations can be found in Appendix B.2.

The error-state Kalman filter maintains a probability distribution $\mathrm{Bel}(\delta \boldsymbol{x}_t)$ that represents the distribution of possible error-states. Recall from Section 2.3.1 that this distribution is assumed to be Gaussian distributed and hence is represented by its mean $\delta \hat{\boldsymbol{x}}_t$ and covariance $\boldsymbol{P}_t$.

3.4.2 Error-state propagation

The Kalman filter propagates its belief of the error-state whenever a new measurement $(\boldsymbol{\omega}_t^b, \boldsymbol{f}_t^b)$ is received from the IMU. The standard Kalman filter update equation (see Appendix C) that is used to propagate the mean error-state is given by

$$\delta \hat{\boldsymbol{x}}_t^- = \boldsymbol{F}_t \delta \hat{\boldsymbol{x}}_{t-\delta t} \tag{3.54}$$

however for the error-state filter it is always the case that $\delta \hat{\boldsymbol{x}}_{t-\delta t} = \boldsymbol{0}$ when entering the propagation step (because the mean error-state is reset to zero when it is transferred to the underlying INS immediately after each correction step). Hence $\delta \hat{\boldsymbol{x}}_t^- = \boldsymbol{0} = \delta \hat{\boldsymbol{x}}_{t-\delta t}$ and so it is not necessary to explicitly update the mean. It is however necessary to propagate the error-state covariance. This is done using the standard Kalman filter update equation

$$\boldsymbol{P}_t^- = \boldsymbol{F}_t \boldsymbol{P}_{t-\delta t} \boldsymbol{F}_t^T + \boldsymbol{Q}_t \tag{3.55}$$

where $\boldsymbol{F}_t$ is the error-state transition matrix and $\boldsymbol{Q}_t$ is the process noise covariance. The error-state transition matrix is a simplified[4] version of the one used in [14]. It contains both first-order and small angle approximations, however these are almost certainly valid at the $100\,\mathrm{Hz}$ sampling rate used. It is given by

$$\boldsymbol{F}_t = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \delta t \boldsymbol{S}(\boldsymbol{f}_t^g) & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \delta t \boldsymbol{I} & \boldsymbol{I} \end{pmatrix} \tag{3.56}$$

where $\boldsymbol{I}$ is the 3×3 identity matrix, $\boldsymbol{0}$ is the 3×3 matrix containing all zeros, δt is the elapsed time since the previous update (e.g. $\delta t = 0.01\,\mathrm{s}$ for an IMU sampling at $100\,\mathrm{Hz}$) and

$$\boldsymbol{S}(\boldsymbol{f}_t^g) = \begin{pmatrix} 0 & -f_t^{g_z} & f_t^{g_y} \\ f_t^{g_z} & 0 & -f_t^{g_x} \\ -f_t^{g_y} & f_t^{g_x} & 0 \end{pmatrix} \tag{3.57}$$

[4]The filter in [14] contains additional accelerometer and gyroscope bias states, which are propagated by the transition matrix into velocity and orientation errors respectively. These additional states were not found to be necessary for the filter's usage in this thesis.

is the skew-symmetric cross-product operator matrix formed from the current specific force vector $\boldsymbol{f}_t^g = (f_t^{g_x}, f_t^{g_y}, f_t^{g_z})$. The $\delta t \boldsymbol{S}(\boldsymbol{f}_t^g)$ sub-block of $\boldsymbol{F}_t$ causes errors in orientation to propagate into velocity errors in the local frame of reference. Errors in velocity are subsequently propagated into displacement errors by the $\delta t \boldsymbol{I}$ sub-block in the third row and second column of $\boldsymbol{F}_t$. The filter uses the error propagation information that is encoded in the transition matrix to track correlations between errors in different components of the state, storing them in the covariance matrix $\boldsymbol{P}_t$. As a result correlated errors in all components of the state are corrected whenever a measurement is received that observes only one of them.

The process noise covariance matrix $\boldsymbol{Q}_t$ describes the error characteristics of the underlying inertial sensors. Its general form is given by

$$\boldsymbol{Q}_t = \begin{pmatrix} \boldsymbol{T}_t & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_t & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \tag{3.58}$$

in which $\boldsymbol{T}_t$ and $\boldsymbol{U}_t$ model gyroscope and accelerometer errors respectively, as described below.

3.4.2.1 Modelling random noise

Random noise is modelled as white noise consisting of a sequence of uncorrelated, Gaussian distributed random variables that perturb the measurements received from the IMU. For the gyroscopes this noise process can be represented exactly by

$$\boldsymbol{T}_t = \delta t^2 \left(\hat{\boldsymbol{C}}_t^-\right)^T (\boldsymbol{W}_t) \left(\hat{\boldsymbol{C}}_t^-\right) \tag{3.59}$$

in which

$$\boldsymbol{W}_t = \begin{pmatrix} (\sigma_\omega^{b_x})^2 & 0 & 0 \\ 0 & (\sigma_\omega^{b_y})^2 & 0 \\ 0 & 0 & (\sigma_\omega^{b_z})^2 \end{pmatrix} \tag{3.60}$$

is a matrix containing the variances of the random variables perturbing measurements from each of the three gyroscopes. White noise perturbing the acceleration measurements can be represented similarly in $\boldsymbol{U}_t$. The variances required in both matrices can be measured directly from signals obtained from an IMU. This can be done using the Allan Variance technique described in Appendix A.

3.4.2.2 Modelling bias instabilities

One way in which bias instability effects can be modelled is to extend the error-state Kalman filter to estimate additional gyroscope and accelerometer bias states [14]. Bias instability effects can then be modelled as random walks in which each bias state is perturbed by additive Gaussian noise, whilst initial uncertainty in the sensor biases can be modelled in $\boldsymbol{P}_0$. Extending the filter in this way was not found to be necessary for the pedestrian dead reckoning filter that is developed in Chapter 4 (and which motivates the development of the error-state filter here) and hence is not described further. A simpler alternative is to use white noise in order to approximate the uncertainty that arises as a result of such errors. Although this approach is theoretically unsound, it achieves good results in practice whilst avoiding the need to extend the state vector (and hence avoiding an increase in the filter's computational requirements). It is preferable to simply ignoring the error sources because underestimating the uncertainty in the state of the underlying INS can limit the filter's ability to apply corrections of sufficient magnitude.

3.4.2.3 Modelling calibration errors

Sensor calibration errors (i.e. non-linearities and scale factor errors) are systematic errors that are not naturally described in terms of additive Gaussian noise. Nevertheless it is helpful to model such error sources so that the filter does not underestimate the amount of uncertainty in the state. Again this can be done by adding additional white noise, whose magnitude results in uncertainty that is roughly equivalent to that expected from the error sources themselves. For example Foxlin used this approach to model non-linearities and temperature dependent scale factor variations in gyroscope signals [80]. A similar approach is used here, defining $\boldsymbol{T}_t$ to be as in Equation 3.59, but with

$$\boldsymbol{W}_t = \begin{pmatrix} (\sigma_\omega^{b_x})^2 + (s_\omega^{b_x}\omega_t^{b_x})^2 & 0 & 0 \\ 0 & (\sigma_\omega^{b_y})^2 + (s_\omega^{b_y}\omega_t^{b_y})^2 & 0 \\ 0 & 0 & (\sigma_\omega^{b_z})^2 + (s_\omega^{b_z}\omega_t^{b_z})^2 \end{pmatrix}. \tag{3.61}$$

The first term in each diagonal component models random noise as described in Section 3.4.2.1 and can be boosted to model otherwise un-modelled error sources as described in Section 3.4.2.2. The second terms model both gyroscope non-linearities and scale factor errors as white noise that is varied dynamically based on the measured angular velocity $\boldsymbol{\omega}_t^b = (\omega_t^{b_x}, \omega_t^{b_y}, \omega_t^{b_z})$. The result is that uncertainty grows more rapidly when the IMU is rotating compared to when it is stationary. This is intended behaviour, since calibration errors are only visible in non-zero measurements. The vector $\boldsymbol{s}_\omega^b = (s_\omega^{b_x}, s_\omega^{b_y}, s_\omega^{b_z})$ determines the magnitude of the resulting uncertainty. Calibration errors perturbing measurements from the

accelerometers are modelled similarly in $\boldsymbol{U}_t$.

The filter implementation used in this thesis models random noise and calibration errors in the way described above. This is done for both the gyroscope and the accelerometer signals. Suitable σ values for modelling random noise were determined for the Xsens Mtx IMU using the Allan Variance technique, as described in Appendix A. The values obtained can be found in Table A.3[5], however they were (somewhat arbitrarily) doubled for use in the filter in order to account for otherwise un-modelled error sources. The vectors $\boldsymbol{s}_\omega^b = (0.001, 0.001, 0.001)$ and $\boldsymbol{s}_a^b = (0.002, 0.002, 0.002)$ were set according to the manufacturer's sensor linearity specifications.

3.4.3 Error-state correction

If an external measurement is made at time t then it is used to correct the estimated error-state. The filter is able to incorporate any measurement $\boldsymbol{z}_t$ that can be related to the true state $\boldsymbol{x}_t$ of the IMU by an equation of the form

$$\boldsymbol{z}_t = \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t) + \boldsymbol{v}_t \,. \tag{3.62}$$

where $\boldsymbol{\mathcal{H}}_t$ is a differentiable (but possibly non-linear) function and $\boldsymbol{v}_t = \mathcal{N}_{0,\boldsymbol{R}_t}$. To apply such a measurement it is first converted into the equivalent error-state measurement $\boldsymbol{y}_t$, given by

$$\boldsymbol{y}_t = \boldsymbol{z}_t - \boldsymbol{\mathcal{H}}_t(\hat{\boldsymbol{x}}_t^-) \tag{3.63}$$

where $\hat{\boldsymbol{x}}_t^-$ is the current inertial navigation solution as estimated by the underlying INS. The minus superscript is used to indicate that the estimate has not yet been corrected by the external measurements. The error-state measurement is used to calculate the posterior belief $(\delta\hat{\boldsymbol{x}}_t, \boldsymbol{P}_t)$ using

$$\boldsymbol{K}_t = \boldsymbol{P}_t^- \boldsymbol{H}_t^T (\boldsymbol{H}_t \boldsymbol{P}_t^- \boldsymbol{H}_t^T + \boldsymbol{R}_t)^{-1} \tag{3.64}$$

$$\delta\hat{\boldsymbol{x}}_t = \boldsymbol{K}_t \boldsymbol{y}_t \tag{3.65}$$

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t) \boldsymbol{P}_t^- \tag{3.66}$$

where $\boldsymbol{R}_t$ is the covariance matrix describing uncertainty in the measurement and

$$\boldsymbol{H}_t = \left[\frac{\partial \boldsymbol{\mathcal{H}}_t}{\partial \boldsymbol{x}} \right]_{\hat{\boldsymbol{x}}_t^-} . \tag{3.67}$$

[5]Note that it is necessary to convert the gyroscope σ values into $\mathrm{rad/s}$ before they can be used in $\boldsymbol{W}_t$.

These are the standard Kalman filter correction equations (see Appendix C) except for 3.65, which is a simplified version of the usual equation that arises for the error-state filter because it is always the case that $\delta\hat{\boldsymbol{x}}_t^- = \mathbf{0}$ when entering the correction step.

In practice it will usually be the case that the frequency at which measurements arrive from the IMU is far greater than the frequency at which external measurements are received. Hence there will be time-steps at which no external measurements are available. In this case the posterior covariance $\boldsymbol{P}_t$ at time t is simply equal to the prior $\boldsymbol{P}_t^-$ as calculated by the propagation step. It may also be the case that more than one external measurement is made within a single time-step. In this case the correction step can be executed multiple times, transferring the error-state to the underlying INS between each execution and using the posterior covariance $\boldsymbol{P}_t$ from one step as the prior covariance $\boldsymbol{P}_t^-$ in the next.

3.4.4 Error-state transfer

Immediately after each correction step the estimated error-state $\delta\hat{\boldsymbol{x}}_t$ is transferred to the underlying INS. Errors in velocity and displacement are transferred by straightforward addition

$$\hat{\boldsymbol{v}}_t = \hat{\boldsymbol{v}}_t^- + \delta\hat{\boldsymbol{v}}_t \tag{3.68}$$

$$\hat{\boldsymbol{s}}_t = \hat{\boldsymbol{s}}_t^- + \delta\hat{\boldsymbol{s}}_t\,. \tag{3.69}$$

The rotation defined by the error vector $\delta\hat{\boldsymbol{\phi}}_t = (\delta\hat{\phi}_t^{g_x}, \delta\hat{\phi}_t^{g_y}, \delta\hat{\phi}_t^{g_z})$ is converted into its equivalent rotation matrix form

$$\hat{\boldsymbol{\Phi}}_t = \begin{pmatrix} 1 & \delta\hat{\phi}_t^{g_z} & -\delta\hat{\phi}_t^{g_y} \\ -\delta\hat{\phi}_t^{g_z} & 1 & \delta\hat{\phi}_t^{g_x} \\ \delta\hat{\phi}_t^{g_y} & -\delta\hat{\phi}_t^{g_x} & 1 \end{pmatrix} \tag{3.70}$$

in which small angle approximations are used. An exact form can be found in Appendix B.2, however in practice the difference between the two is negligible provided that frequent corrections are applied. Having calculated $\hat{\boldsymbol{\Phi}}_t$, the corrected INS attitude is given by

$$\hat{\boldsymbol{C}}_t = \hat{\boldsymbol{\Phi}}_t \hat{\boldsymbol{C}}_t^-\,. \tag{3.71}$$

Once the mean error-state has been transferred to the underlying INS it is reset to zero.

An outline of the complete system is shown in Figure 3.5. The downward arrows from the error-state Kalman filter represent the transfer of the estimated error-state to the INS.

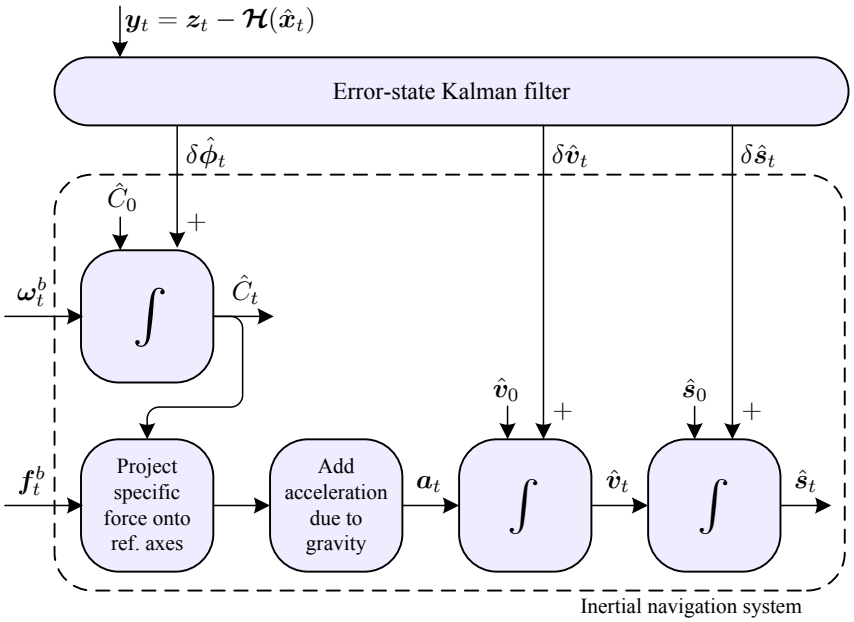


Figure 3.5: INS correction using an error-state Kalman filter.

3.4.5 Xsens Mtx example

One way to test the performance of the error-state Kalman filter is to revisit the trials conducted with the Xsens Mtx IMU in Section 3.3.1. Tracking this IMU using an uncorrected INS resulted in an average drift of $1052\,\mathrm{m}$ after $120\,\mathrm{s}$. In this section the same data is processed, but the INS is periodically corrected using external measurements. In particular it is assumed that the device is known to be approximately stationary for $0.5\,\mathrm{s}$ periods uniformly spaced at $2\,\mathrm{s}$ intervals from $t = 0\,\mathrm{s}$ to $t = 120\,\mathrm{s}$ of each run. When a new sample is received from the IMU during these periods a corresponding zero velocity update (ZVU) is generated and is used as a pseudo-measurement by the error-state filter in order to correct the INS. Note that this hypothetical test case is similar to the application of ZVUs in pedestrian dead reckoning systems, as described in Section 2.2.2.2.

To illustrate the benefit of the error-state filter, the measurements were applied both naïvely (i.e. just correcting the velocity component of the INS state) and using the filter. In both cases the INS state was initialised in the same way as in Section 3.3.1. In the latter case the required measurement function and its derivative are given by

$$\boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t) = \boldsymbol{v}_t \tag{3.72}$$

$$\boldsymbol{H}_t = (\boldsymbol{0}, \boldsymbol{I}, \boldsymbol{0})\,. \tag{3.73}$$

Each ZVU was assumed to have a small amount of uncertainty that was independent along

each axis. Hence the measurement covariance matrix was set to

$$\boldsymbol{R}_t = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{pmatrix} \tag{3.74}$$

in which a value of $\sigma = 0.01\,\mathrm{ms}^{-1}$ was used. The filter was initialised assuming that there was no uncertainty in the initial displacement and heading, and a small amount of uncertainty in the tilt and velocity of the device. Hence the initial state covariance was

$$\boldsymbol{P}_0 = \begin{pmatrix} \boldsymbol{V} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{3.75}$$

where

$$\boldsymbol{V} = \begin{pmatrix} 10^{-3} & 0 & 0 \\ 0 & 10^{-3} & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.76}$$

$$\boldsymbol{W} = \begin{pmatrix} 10^{-5} & 0 & 0 \\ 0 & 10^{-5} & 0 \\ 0 & 0 & 10^{-5} \end{pmatrix} \tag{3.77}$$

with units $(\mathrm{rad/s})^2$ and $(\mathrm{m/s})^2$ respectively. Note that in general it may be necessary to initialise the other diagonal elements of $\boldsymbol{P}_0$ to be small positive numbers (rather than zeros) to ensure that $\boldsymbol{P}_t$ remains positive definite. This is a necessary condition to prevent filter instabilities and likely divergence of the system away from the true state [53]. Appropriate values depend on the floating point implementation used. In practice no instabilities were observed when zeros were used in the tested implementation, which was written in Java using 64-bit floating point arithmetic.

Figure 3.6 shows the mean drift in position (averaged over the same 100 event streams as in Section 3.3.1) when the INS was uncorrected, naïvely corrected and corrected using the error-state filter. The mean drift when naïve corrections were applied is shown together with error bars in Figure 3.7. The drift was significantly reduced to an average of $14.3\,\mathrm{m}$ after $120\,\mathrm{s}$, however since the orientation component of the state is left uncorrected drift still accumulates at a rate that is proportional to t^2 after gyroscope errors become the dominant source of drift. This is an improvement on the unconstrained INS in which drift grew proportional to t^3 during the same period, however it is possible to do much better. Figure 3.8 shows the mean drift when the ZVU measurements were applied using the error-state filter. Note that the mean drift of $0.04\,\mathrm{m}$ after $120\,\mathrm{s}$ is much lower than when naïve corrections are applied, and that drift now

grows at a rate that is roughly linear with respect to time. This improvement is a result of the filter's ability to track correlations between errors that accumulate in different components of the INS state. By using this information the filter is able to correct not only errors in velocity when applying ZVUs, but also tilt errors in the tracked orientation and much of the drift in position that has accumulated since ZVUs were last applied. These corrections are visible in Figure 3.8 as sharp reductions in the average drift that correspond to regions of time during which ZVUs were applied.

3.5 Conclusions

Inertial navigation can be used to track the position, velocity and orientation of an IMU relative to its initial state, however drift errors rapidly accumulate in the tracked position. Even when using a high-end MEMS IMU, drift will typically exceed $170\,\mathrm{m}$ after one minute of operation. Furthermore drift grows proportional to t^3 (after the point in time at which the gyroscopes become the dominant source of error), meaning that huge improvements in the quality of MEMS inertial sensors are required to achieve only modest increases in the duration for which objects can be tracked to a specified level of accuracy. Hence unconstrained inertial navigation is not suitable for pedestrian tracking and this will remain the case for the foreseeable future. Many indoor location-aware applications require metre-level accuracy over extended periods of time, however even with a perfect IMU drift will exceed $1\,\mathrm{m}$ after less than one hour unless local gravitational fluctuations are taken into account (see Section 2.2). Technologies that compensate for such fluctuations will need to be developed in addition to small and lightweight inertial sensors that give better than strategic-grade levels of performance if this requirement is ever to be met.

Although unconstrained inertial navigation cannot be used to track the position of a pedestrian, constrained inertial navigation in which external measurements and known constraints are used to frequently correct an INS can be used. Applying such measurements using an error-state Kalman filter is superior to directly correcting the state because such filters are able to track correlations between errors that accumulate in different components of the state. As a result it is possible to correct errors in components of the state that are not observed directly by the measurements. For example zero velocity measurements can be used to correct the velocity, tilt and (to some extent) position components of the state of an INS.

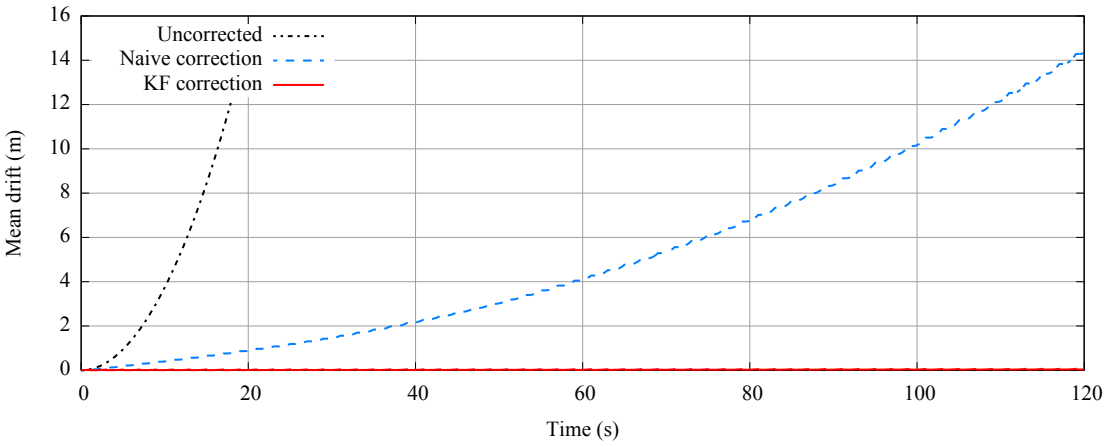


Figure 3.6: The mean drift incurred by an INS when uncorrected, naïvely corrected by ZVUs, and corrected by ZVUs using the error-state Kalman filter.

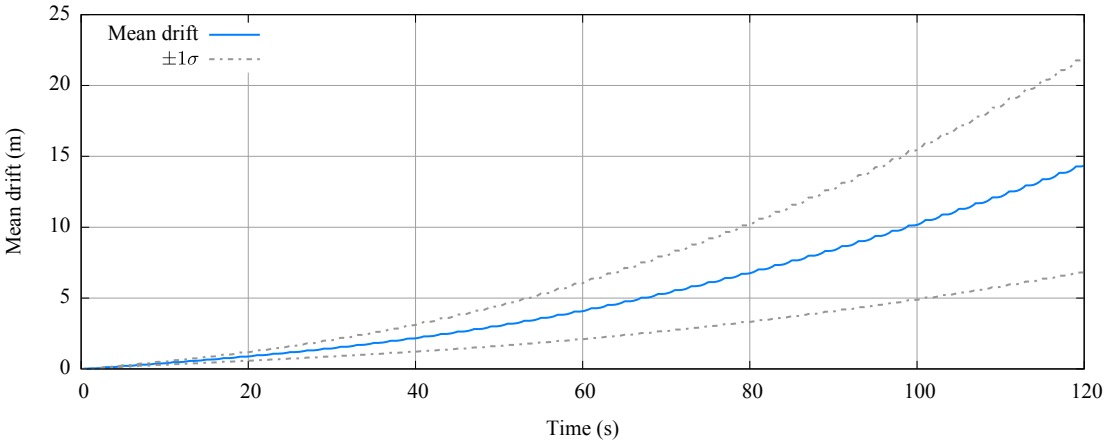


Figure 3.7: The mean drift incurred when the INS was corrected using the naïve method.

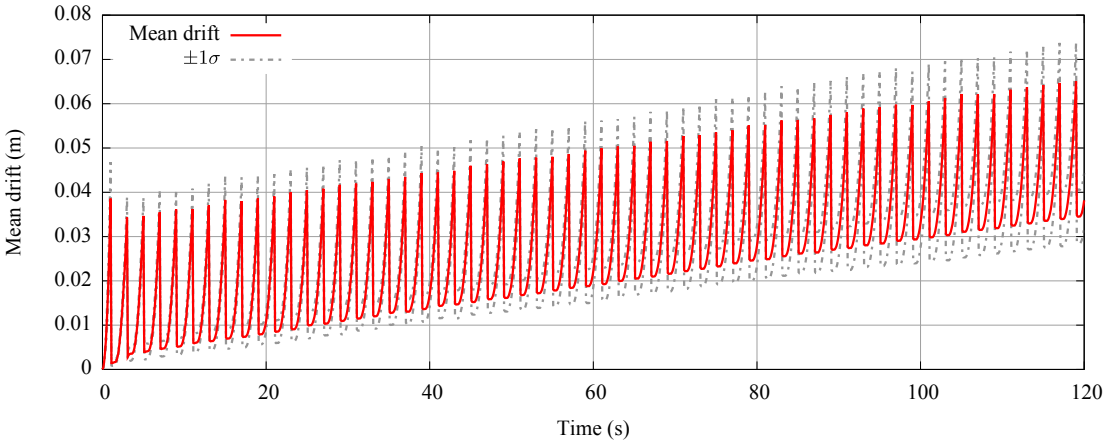


Figure 3.8: The mean drift incurred when the INS was corrected using the error-state Kalman filter.

Chapter 4

Pedestrian dead reckoning

This chapter describes the development of a pedestrian dead reckoning (PDR) filter that makes use of domain specific knowledge in order to address the issues with unconstrained inertial navigation that were identified in Chapter 3. The approach used is similar to that of existing inertial navigation based PDR systems such as those described in Section 2.2.2.2, however the intended use of the PDR filter to provide measurements to a higher level localisation filter (see Figure 1.1 on page 5) results in some small but important differences. As well as highlighting these differences, this chapter evaluates the performance of the filter and describes the format of the measurements that it generates for consumption by the higher level filter.

4.1 INS-based PDR

An INS that is corrected using the error-state Kalman filter described in Section 3.4 forms the basis of the PDR filter described in this section. It is necessary to provide frequent external measurements to the Kalman filter so that it is able to reduce the rate at which drift accumulates in the state of the underlying INS. This is achieved by using a foot-mounted IMU and applying zero velocity update (ZVU) pseudo-measurements when the user's foot is known to be stationary. An algorithm for detecting when the foot is stationary and applying ZVUs is described in Section 4.1.1.

Whilst previous PDR systems have been designed primarily for standalone use, the filter developed in this chapter is designed specifically to generate relative movement events for use by a higher level localisation filter, which is described in Chapter 5. This higher level filter combines relative movement measurements with environmental constraints such as walls in order to determine and subsequently track the user's absolute position. The requirements of the higher level filter result in a number of differences between existing standalone PDR systems and the one that is developed here. Firstly, relative movement events must be generated

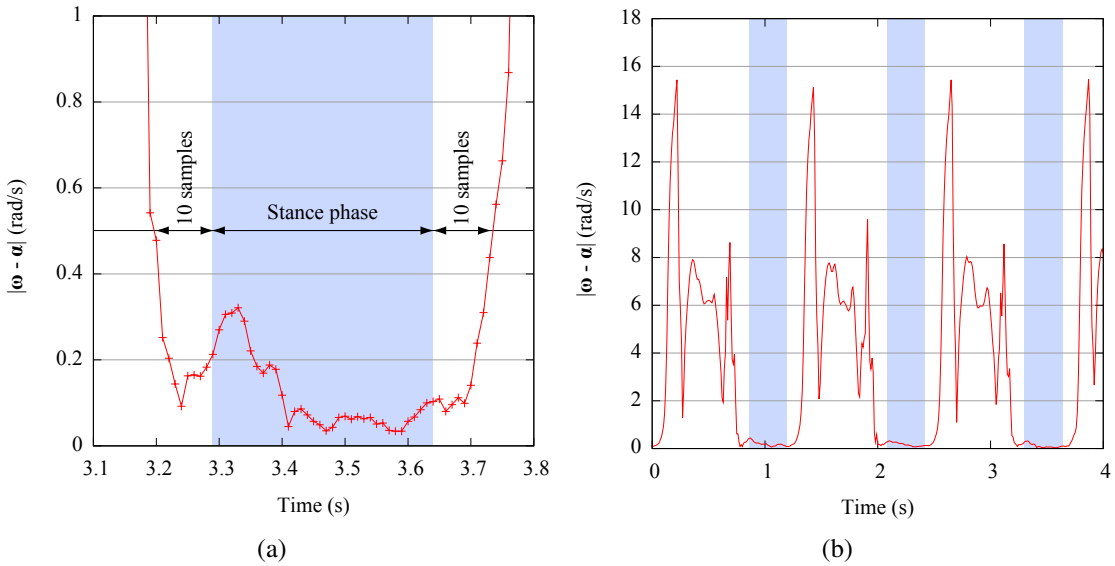


Figure 4.1: The magnitude of the angular velocity (a) during a single stance phase, and (b) during three steps. The shaded regions indicate the detected stance phases within which ZVUs are applied.

at a frequency that is much lower than the sampling frequency of the IMU. This is because the update that is performed by the localisation filter whenever a new event is received is computationally expensive and hence can only be performed at a relatively low frequency. Hence it is necessary to segment the user's movement into coarse events. An approach in which the user's movement is divided into coarse events corresponding to steps taken by the user is described in Section 4.1.2. Secondly, unlike in standalone PDR systems, it is not always desirable to track the user's position as accurately as possible when generating step events for use by a localisation filter. In particular it is not desirable to generate step events that contain corrections for errors in previously generated events. Section 4.1.3 describes why this is the case and presents a simple way in which the generation of such events can be prevented.

4.1.1 Zero velocity updates

In order to apply ZVU pseudo-measurements it is necessary to identify periods of time during which the user's foot is stationary (called stance phases). This can be done simply by detecting when the magnitude of the angular velocity falls below a certain threshold. The following algorithm has been found to reliably detect stance phases.

- A stance phase starts $0.1\,\mathrm{s}$ (10 samples) into a period of time during which the bias-corrected angular velocity has remained below a threshold of 0.5:

$$|\boldsymbol{\omega}_t^b - \hat{\boldsymbol{\alpha}}_t| < 0.5\,\mathrm{rad/s}\,. \tag{4.1}$$

The delay ($0.1\,\mathrm{s}$) and threshold (0.5) values were empirically determined to give reliable results.

- A stance phase ends $0.1\,\mathrm{s}$ (10 samples) before the first sample which exceeds the threshold:

$$|\boldsymbol{\omega}_t^b - \hat{\boldsymbol{\alpha}}_t| \geq 0.5\,\mathrm{rad/s}\,. \tag{4.2}$$

  Note that this condition requires samples from the IMU to be delayed by $0.1\,\mathrm{s}$ during each stance phase before they can be passed to the INS's navigation processor. This delay has a negligible effect on the output of the filter, since the IMU is (by definition) near-stationary throughout each stance phase.

The detection of a single stance phase is illustrated in Figure 4.1(a). The stance phases detected for three steps taken by a user are shown in Figure 4.1(b).

ZVUs are generated and applied throughout each stance phase. The measurement vector and model that are required by the filter are given by

$$\boldsymbol{z}_t = (0, 0, 0)^T \tag{4.3}$$

$$\boldsymbol{H}_t = (\mathbf{0}, \boldsymbol{I}, \mathbf{0})\,. \tag{4.4}$$

When a ZVU is applied it is usually the case that the user's foot is not perfectly stationary. This uncertainty is modelled in the measurement covariance matrix $\boldsymbol{R}_t$, in which it is assumed that any residual velocity occurs independently along each axis and has variance $\sigma^2 = 0.01\,(\mathrm{ms}^{-1})^2$, which was empirically determined to give good performance.

4.1.2 Step segmentation

The INS updates an estimate of the 3-dimensional position and orientation of the user's foot (relative to a specified starting position and orientation) 100 times per second. In practice few systems or applications require updates at such a high frequency (with the notable exceptions of virtual reality and motion capture applications). In particular the localisation filter that will be described in Chapter 5 requires events describing the user's relative movement to be reported at a much lower frequency (in the order of one event per second[1]). To provide such events the relative movement is segmented into *step events*[2] of the form

$$\boldsymbol{e}_t = (l_t, \delta z_t, \delta\theta_t, \xi_t) \tag{4.5}$$

[1]The exact frequency at which events can be processed by the higher level filter is dependent on the size of the building in which the user is tracked, the computational resources that are available and the degree of certainty to which the user's absolute position is known.

[2]This term is slightly misleading. Each step event actually corresponds to two steps taken by the user because the filter only detects steps taken by one of the user's feet.

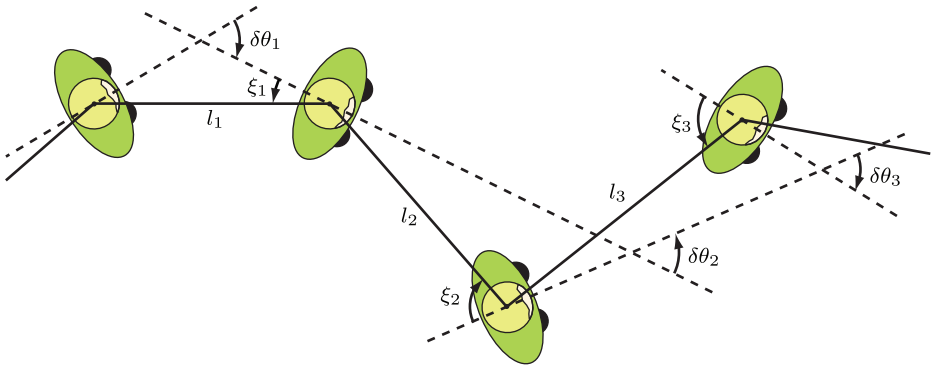


Figure 4.2: The segmentation of a pedestrian's relative movement into step events. The parameters (excluding δz_t) corresponding to three step events are shown.

corresponding to steps taken by the foot to which the IMU is attached. A step event parameterises the relative movement of the user's foot during a step, describing the movement in terms of its length l_t in the horizontal (x, y) plane, the change in vertical displacement δz_t, the change in the user's heading $\delta\theta_t$ and the offset ξ_t between the user's final heading and the direction of the step, as illustrated by Figure 4.2. Note that this parametrisation specifies the user's heading independently from the direction of the step. Hence a localisation filter receiving such events can distinguish between a $90°$ turn and a sidestep. As a result such a filter can correctly estimate the user's heading (i.e. the direction in which they are facing) in each case.

Step events are generated at the end of each stance phase. This ensures that the state of the INS has been corrected as much as is possible by the error-state Kalman filter immediately prior to the generation of each event. The one case in which this approach is not ideal is when the user stops walking. In this case the event corresponding to the user's last step will not be reported until he or she resumes walking, since only then will the corresponding stance phase end. In practice this is easily solved by generating a step event part way through any stance phase that is found to exceed a specified duration. Note that a step event generated part way through a stance phase is generated instead of, rather than in addition to, an event at the end of the same phase. Generating step events at most $0.5\,\mathrm{s}$ into each stance phase was found to work well in practice. Figure 4.3 shows the points in time at which step events corresponding to five steps are generated. The parameters of a step event are derived from the state of the underlying INS together with the state when the previous step event was generated. The algorithm used to generate each step event is given by Algorithm 1, in which it is assumed that the user's heading is aligned with the x-axis of the IMU (therefore the IMU should be mounted on the foot such that this is the case). Note however that a mounting misalignment will not result in any errors in the path of step events. It will only cause each ξ_t to be offset from its true value. In other words the step events will be the same as if the IMU was correctly aligned, but the user's heading will be reported as though he or she is walking a little to one side when they are in fact walking forwards.

Algorithm 1 GenerateStep - Generates a step event from the current state of the underlying INS and its state at time $(t - \delta t)$ when the previous step event was generated. Square brackets are used to index into vector quantities, with the first element in each vector assigned an index of zero.

procedure GENERATESTEP($(\hat{\boldsymbol{C}}_t, \hat{\boldsymbol{s}}_t, \hat{\boldsymbol{v}}_t)$, $(\hat{\boldsymbol{C}}_{t-\delta t}, \hat{\boldsymbol{s}}_{t-\delta t}, \hat{\boldsymbol{v}}_{t-\delta t})$)
    // Calculate the change in displacement
    $\delta\hat{\boldsymbol{s}} \leftarrow \hat{\boldsymbol{s}}_t - \hat{\boldsymbol{s}}_{t-\delta t}$
    // Assume that the user's heading ($\boldsymbol{h}$) is aligned with the x-axis of the IMU
    $\boldsymbol{h}^b \leftarrow (1, 0, 0)^T$
    // Calculate $\boldsymbol{h}$ in the local frame at the start ($\boldsymbol{h}^g_{t-\delta t}$) and end ($\boldsymbol{h}^g_t$) of the step
    $\boldsymbol{h}^g_{t-\delta t} \leftarrow \hat{\boldsymbol{C}}_{t-\delta t}\boldsymbol{h}^b$
    $\boldsymbol{h}^g_t \leftarrow \hat{\boldsymbol{C}}_t\boldsymbol{h}^b$
    // Calculate the change in the user's heading
    $\delta\theta_t \leftarrow \text{atan2}(\boldsymbol{h}^g_t[1], \boldsymbol{h}^g_t[0]) - \text{atan2}(\boldsymbol{h}^g_{t-\delta t}[1], \boldsymbol{h}^g_{t-\delta t}[0])$
    // Calculate the step offset
    $\xi_t \leftarrow \text{atan2}(\boldsymbol{h}^g_t[1], \boldsymbol{h}^g_t[0]) - \text{atan2}(\delta\hat{\boldsymbol{s}}[1], \delta\hat{\boldsymbol{s}}[0])$
    // Calculate the step length and vertical displacement
    $l_t \leftarrow \sqrt{\delta\hat{\boldsymbol{s}}[0] * \delta\hat{\boldsymbol{s}}[0] + \delta\hat{\boldsymbol{s}}[1] * \delta\hat{\boldsymbol{s}}[1]}$
    $\delta z_t \leftarrow \delta\hat{\boldsymbol{s}}[2]$
    // Return the step event
    return $(l_t, \delta z_t, \delta\theta_t, \xi_t)$
end procedure

By concatenating a sequence of step events generated by the PDR filter it is possible to reconstruct the relative movement of the user as calculated by the INS, albeit without the fine-grained motion of the foot during each step. An example of the fine-grained motion calculated by the INS (with ZVU corrections applied by the error-state Kalman filter) and the path obtained by concatenating the corresponding step events is shown in Figure 4.4. Note that the paths have been manually aligned with one other and with a model of the stairs that the user was descending when the data were obtained.

4.1.3 The cascading filter problem

The error-state Kalman filter that is used by the PDR filter is able to track growing uncertainty in the state of the underlying INS over multiple steps. In theory this can cause an error in position to grow slowly over multiple steps, before being reduced rapidly at the end of a further step when filter is able to resolve much of the accumulated uncertainty. For a standalone PDR system this is a desirable property, since the filter is able to correct errors that have accumulated during previous steps and hence return to a more accurate estimate of the user's position. In contrast such corrections are not desirable when the filter is used to generate step event measurements for a localisation filter.

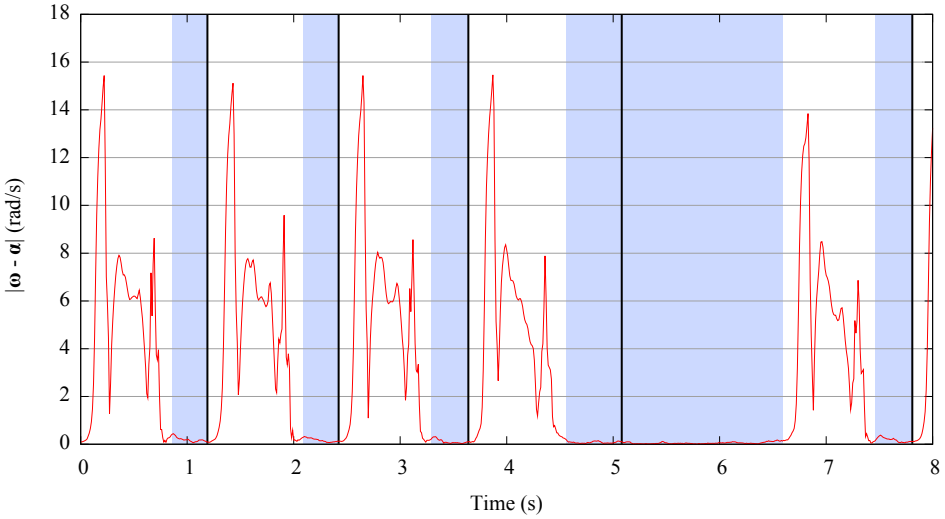


Figure 4.3: A step event is generated $0.5\,\mathrm{s}$ into a stance phase, or at the end of a stance phase lasting less than $0.5\,\mathrm{s}$. Each stance phase is indicated by a shaded region. Vertical black lines indicate points in time at which step events are generated.

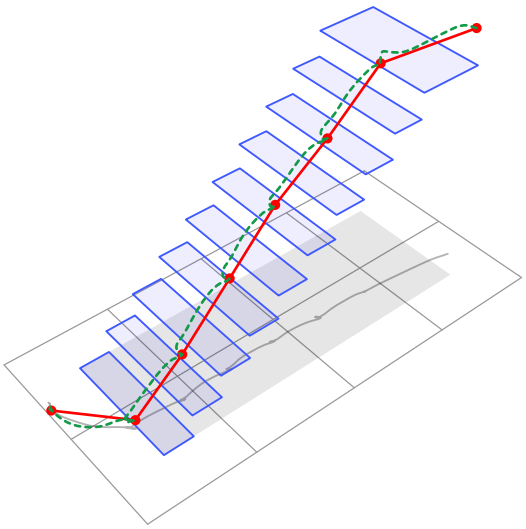


Figure 4.4: The fine-grained motion calculated by the INS (with ZVU corrections) as a pedestrian walked down a flight of stairs (dashed green line), and the path of step events generated by the PDR filter (solid red line). A dot indicates the start of each step event. Grid size = $1\,\mathrm{m}^2$.

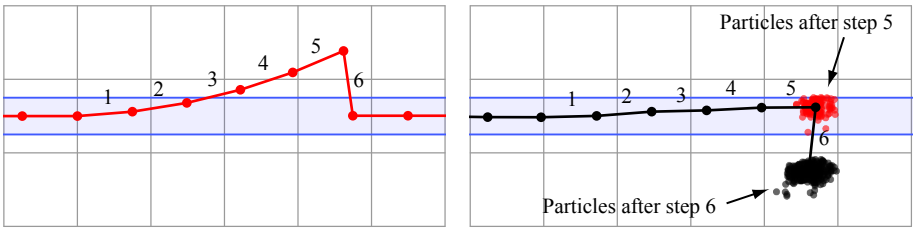


Figure 4.5: The problem of delayed drift correction. (a) A sequence of step events in which the sixth step corrects drift that has accumulated during the previous five events. (b) The corrected path calculated by a localisation filter. A failure occurs when the localisation filter attempts to process the 6^{th} step.

Consider the hypothetical case shown in Figure 4.5, in which a user is walking down the centre of a straight corridor (shown from above). A sequence of step events that might be generated by a PDR filter is shown in Figure 4.5(a), in which drift accumulates over five step events before being corrected in a sixth. A localisation filter receiving these step events will use the walls of the corridor to correct the drift reported in each of the first five step events, as shown in Figure 4.5(b). The sixth step event contains a large error that compensates for drift in the previous five events, however the localisation filter has already corrected most of this drift and as a result a localisation failure (where all of the particles that the localisation filter uses to represent the user's position are found to violate environmental constraints) occurs. This is an example of a cascading filter problem in which the second filter (i.e. the localisation filter) does not correctly model the errors in the first filter's outputs.

In practice large corrections such as the one shown in Figure 4.5(a) are rare, since if ZVUs applied during one stance phase are unable to correct drift in position then it is quite unlikely that ZVUs applied during subsequent stance phases will be able to do any better. Nevertheless such behaviour is occasionally observed and therefore must be addressed. There are several existing solutions to the problem of cascading filters. These include the federated zero-reset architecture, in which the state of the first filter is zeroed following input to the second filter, with the corresponding elements of the first filter's covariance matrix reset to their initialisation values [18]. The approach used here is very similar. Whenever a step event is generated the error-state covariance matrix $\boldsymbol{P}_t$ is simply reset to its initial value, as given by Equation 3.75 (page 64). This effectively asserts that the current position and heading estimates are correct (since $\boldsymbol{P}_0$ assumes no uncertainty in these components of the state). Hence it is not possible for the filter to correct errors in previously generated step events. Note that it is not necessary to zero the state of the underlying INS (as in the zero-reset architecture) because each step event is calculated as the difference between the state at the start and end of the corresponding period of time (see Section 4.1.2).

In addition to solving the cascading filter problem, the solution outlined above also prevents

elements of the error-state covariance matrix from growing large over time. If the matrix were not reset then the elements of $\boldsymbol{P}_t$ corresponding to heading and position would grow over time to represent growing uncertainty in the heading (and hence also the position) of the user. This could potentially destabilise the PDR filter if it were used for a sufficiently long period of time. By resetting $\boldsymbol{P}_t$ whenever a step event is generated, this possibility is avoided.

4.1.4 Filter initialisation

The initial state of the PDR filter is initialised in a similar way to that described in Section 3.3.1. The initial velocity and displacement of the underlying INS are both set to zero

$$\hat{\boldsymbol{s}}_0^g = \hat{\boldsymbol{v}}_0^g = (0, 0, 0)^T \tag{4.6}$$

whilst the initial orientation of the IMU is set such that the global gravity vector $\boldsymbol{g}^g = (0, 0, -9.807)^T$ is aligned with the average specific force vector measured by the IMU during the first stance phase. The heading component of the orientation can be initialised to an arbitrary value, since the filter calculates relative changes in the user's heading as opposed to absolute heading estimates. As in Section 3.4.5 it is assumed that there is no uncertainty in the initial state except for a small amount of uncertainty in the tilt of the IMU. The initial covariance matrix $\boldsymbol{P}_0$ is given by Equation 3.75 (on page 64).

4.1.5 Alternative filter formulation

The PDR filter described above is one of many filters capable of generating step events of the form described in Section 4.1.2. Its design is convenient in that it makes use of the standard INS and error-state Kalman filter components developed in Chapter 3. Since the filter does not estimate the components of each step event (i.e. the horizontal length, change in vertical displacement, change in heading and direction offset) directly, it is necessary to perform a conversion to obtain them from the state of the underlying INS. An alternative (and neater) approach would be to develop a Kalman filter that estimates the components of each step event directly. Step events could then be generated directly from the filter's state, without the need for a conversion. After each step event is generated the state of such a filter would be reset to zero and its state covariance would be reset to its initial value, as in the federated zero-reset architecture described in Section 4.1.3. This approach (and other potential approaches) are not explored further in this thesis, however it is worth noting that there are many ways of developing a filter capable of performing the same task as the one described above.

(a) (b)

Figure 4.6: Components of a prototype PDR system. (a) An OQO UMPC. (b) The UMPC was connected by wire to a foot-mounted IMU.

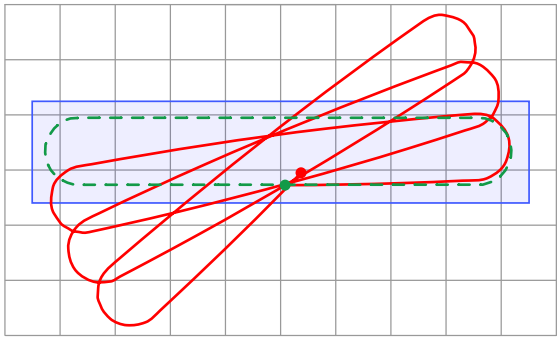


Figure 4.7: The top down view of a closed path followed by a pedestrian (dashed green line) and the path calculated by a hypothetical PDR system (solid red line). Dots indicate the end of each path. The error in position at the end of the path is far smaller than the errors incurred at either end of the corridor.

4.2 Evaluation

A prototype implementation of the PDR filter described in Section 4.1 was developed using an OQO ultra mobile PC (UMPC) running Linux (Figure 4.6(a)) and an Xsens Mtx IMU. The UMPC was carried in the user's pocket and was connected by wire to the IMU (Figure 4.6(b)). Tape was used to attach the IMU to the front of the user's foot during testing. The use of a UMPC allowed the software components of the PDR system to be developed rapidly using the Java programming language, whilst the use of a wire to connect it to the IMU ensured reliable communication between the two devices. Note that in a practical deployment of the filter the UMPC could be replaced by a much cheaper micro-controller that could be embedded into the IMU package on the user's foot.

Figure 4.8: An Active Bat mounted above an Xsens IMU on the user's foot.

4.2.1 Previous testing methodologies

In previous research standalone PDR systems have been evaluated in indoor environments by tracking a pedestrian around a closed path within a building (e.g. [14]). The tracking accuracy is then measured as the difference between the initial and final positions calculated by the system and is often specified as a percentage of the total distance travelled. Unfortunately this approach does not always give a good indication of the true accuracy of a PDR system, particularly if errors in the estimated heading are the main cause of drift in position. Figure 4.7 shows a hypothetical example in which a user is tracked walking up and down a corridor three times, starting and ending at the midpoint of the corridor. Bias errors in the gyroscopes of the hypothetical system cause a heading error that grows linearly over time. This results in large position errors when the user is at either end of the corridor, but the final error in position when the user returns to the midpoint remains small. Hence it can be misleading to use the final error in position as an indication of the system's accuracy. Furthermore the final error in position does not provide any indication of its cause (e.g. in this example errors in position were caused by drift in the estimated heading).

4.2.2 Ground truth

In order to perform a thorough evaluation of the PDR filter, measurements from the Active Bat ultrasonic location system were used to calculate a ground truth to which paths of step events calculated by the filter could be compared. Recall from Section 2.1.2.2 that the Active Bat system measures ranges between mobile tags (known as a *bats*) and multiple receivers installed at known locations in the ceiling. A multi-lateration algorithm is applied to these range measurements to position bats to within $3\,\mathrm{cm}$ 95% of the time [6]. The Active Bat system is currently installed in one wing of the William Gates building (Cambridge, UK) in which the evaluation was conducted.

To calculate a ground truth for the user's movement, a bat was mounted above the IMU on the test subject's foot as shown in Figure 4.8. This bat was queried at a rate of $25\,\mathrm{Hz}$ by the Active Bat system during testing. The PDR filter was extended so that range measurements obtained from the Active Bat system could be incorporated directly into the filter, allowing the construction of a drift free ground truth during each test. Drift does not accumulate in the state estimated by the extended filter because the absolute position of the user's foot is constrained each time range measurements are applied. Furthermore the heading calculated by the extended filter is also drift free. This is because any error in the estimated heading causes a correlated error in position (unless the user remains stationary). The filter tracks such correlations as described in Section 3.4 and hence is able to correct heading errors whenever range measurements are applied.

Fully describing the extension of the filter to incorporate range measurements requires a significant amount of space and is largely irrelevant to the evaluation of the PDR filter itself. A full description can be found in Appendix D, with the main points summarised below.

- The equations used to incorporate range measurements must take into account the offset between the Active Bat and the origin of the IMU (i.e. the point of the IMU that is actually tracked by the INS) in order to achieve accurate results. Suitable equations can be found in Section D.1.

- It was necessary to ensure that measurements from the Active Bat system and the IMU were correctly synchronized. The Chrony[3] time daemon was used to synchronize the UMPC with the machine responsible for logging Active Bat measurements. In addition Active Bat measurements were only applied if they were made during the stance phase. This approach is tolerant to small timing errors because the user's foot is stationary during each stance phase. More details can be found in Section D.4.1.

- It was necessary to detect and filter out range measurements that correspond to reflected (i.e. indirect) ultrasonic pulses. This can be done by comparing range measurements with those predicted by the current state estimate, as described in Section D.4.2.

The result of incorporating range measurements from the Active Bat system is a combined inertial and ultrasonic absolute positioning system that can be expected to position the user's foot at least as accurately as the Active Bat system alone following corrections during each stance phase (i.e. positions calculated at the end of each stance phase should be accurate to within $3\,\mathrm{cm}$ 95% of the time). The ability of the extended filter to also calculate drift free heading estimates is the main benefit of its use as a ground truth over simply using the Active Bat system directly. Due to the lack of a more accurate ground truth it was not possible to determine the accuracy to which the system was able to estimate the user's heading, however

[3] http://chrony.tuxfamily.org

Table 4.1: The application of Active Bat range measurements (AB), zero velocity updates (ZVU) and $\boldsymbol{P}_t$ resets (PR) during test runs.

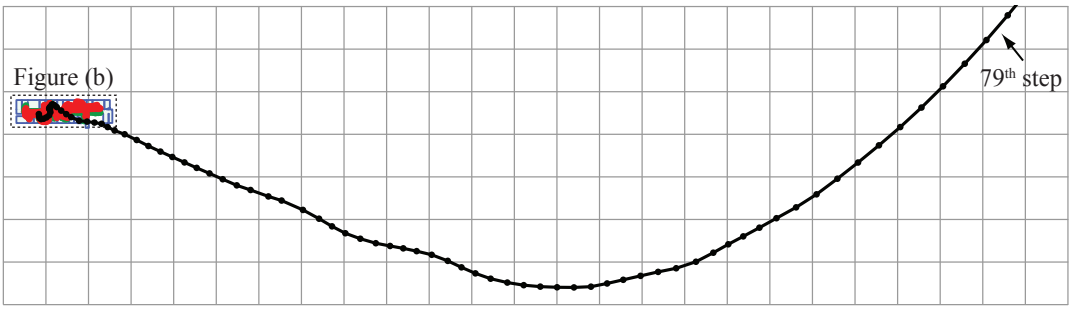
	Initialisation			During run		
	AB	ZVU	PR	AB	ZVU	PR
Unconstrained filter	Yes	Yes	No	No	No	No
PDR filter	Yes	Yes	No	No	Yes	Yes
Ground truth filter	Yes	Yes	No	Yes	Yes	No

simulations for a similar combined inertial and ultrasonic positioning system predict errors of less than 1° [19].

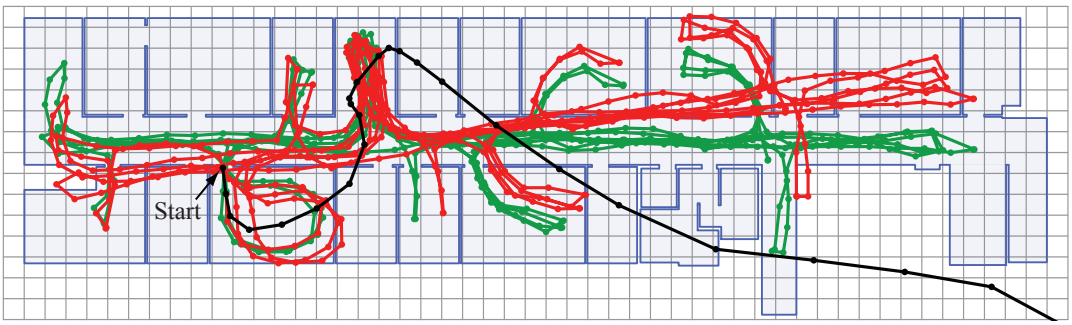
It is important to note that the ground truth filter cannot be used to determine the error in each *individual* step event generated by the PDR filter. This is not only because the Active Bat system is not itself accurate enough to measure the kind of errors that are expected, but also because the ground truth filter processes the same IMU data as the PDR filter itself. Hence correlations will exist between errors incurred in the individual step events calculated by the two filters. The ground truth filter is however suitable for measuring the *accumulation* of drift in both heading and position over time.

During testing the PDR filter was compared to both the ground truth filter and an uncorrected INS. To make such comparisons all three were implemented using the extended version of the PDR filter described above. During each test run the first 15 steps detected by the filter were used as an initialisation phase. During this phase all three filters applied Active Bat range measurements and ZVUs. This ensured that all three filters had the same initial state. Rather than the initial state containing an initial displacement of $\hat{\boldsymbol{s}}_0^g = (0, 0, 0)$ and an arbitrary heading (as is the case in the normal operation of the PDR filter), this approach initialised each filter with an absolute position and heading that was effectively derived using the range measurements obtained from the Active Bat system. This allowed the paths calculated by each filter to be rendered on maps of the environment without any need for manual alignment. During the initialisation phase none of the three filters reset the error-state covariance matrix $\boldsymbol{P}_t$ at the end of each step, as was described in Section 4.1.3. Such resets effectively assert that the current position estimated by the filter is correct, however this is counter-productive when range measurements are themselves being used to correct the estimated position.

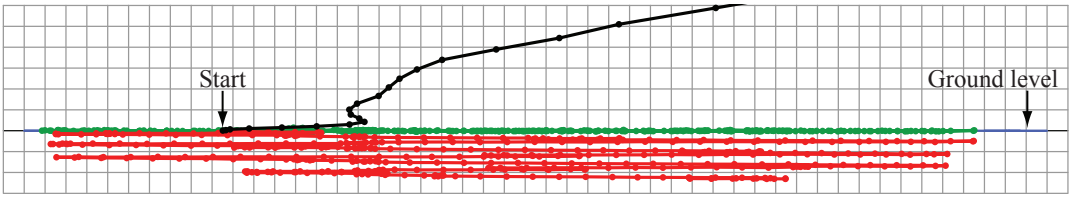
After the initialisation phase each of the three filters applied a different combination of measurements. For the uncorrected INS case no measurements were applied, the PDR filter applied ZVUs and the ground truth filter continued to apply both ZVUs and Active Bat range measurements. In all three cases the movement was segmented into step events as described in Section 4.1.2. Only in the PDR filter case was the error-state covariance $\boldsymbol{P}_t$ reset following the generation of each step event. A summary of the measurements and error-state covariance resets that were applied in each of the three cases is shown in Table 4.1.



(a) Top down view (overview). Grid size $25\,\mathrm{m}^2$.



(b) Top down view (zoomed). Grid size $1\,\mathrm{m}^2$.



(c) Side view. Grid size $1\,\mathrm{m}^2$.

Figure 4.9: Paths of step events generated by unconstrained inertial navigation (black), the PDR fitler (red), and the ground truth filter (green).

4.2.3 Results

The results obtained for a single walk in the area covered by the Active Bat system are shown in Figure 4.9. The initialisation phase in which range measurements were applied by all three filters is not shown. Note that despite the appearance of Figure 4.9, the PDR filter is only capable of tracking the *relative* movements of a pedestrian. In normal operation the PDR filter has no way of determining its initial state and hence is initialised with a position of $(0, 0, 0)$ and an arbitrary heading as described in Section 4.1.4.

For the example in Figure 4.9 the user made 800 steps after the initialisation phase (causing 400 step events to be generated by each of the three filters) over a period of just under nine minutes, walking a distance of approximately $500\,\mathrm{m}$. As expected from the results in Section 3.4.5, the position estimated by the unconstrained INS diverged rapidly from the true position of the user. Note that Figure 4.9(a) only shows the positions calculated by the INS for the first 80 step events. After 400 step events the total drift in position (measured as the Euclidean

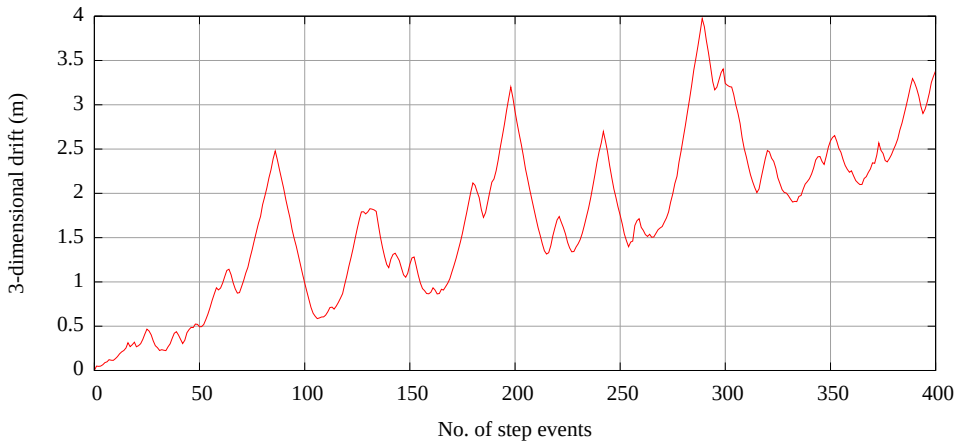


Figure 4.10: The 3-dimensional drift in position (measured as the Euclidean distance between the positions calculated by the PDR filter and the ground truth filter) incurred by the PDR filter during the example shown in Figure 4.9.

distance between the positions estimated by the unconstrained INS and the ground truth filter) was $43.15\,\mathrm{km}$. The PDR filter achieved far better performance, with a final drift of $3.48\,\mathrm{m}$. This can be broken down into drift in the (x, y) plane and drift in the vertical direction. Figure 4.9(c) shows that in this example there seems to be a systematic error that caused downward drift over time, resulting in a final error in vertical displacement of $-2.60\,\mathrm{m}$. Figure 4.9(b) shows that there was also an error in heading, which caused drift in the (x, y) plane that was largest when the user was at either end of the corridor. This is responsible for the peaks in Figure 4.10, which plots the drift incurred by the PDR filter over the duration of the run. The worst case drift at any point during the walk was $3.97\,\mathrm{m}$, however note that in general this value is dependent on the path followed by the user. If the test were repeated whilst the user walked up and down a longer corridor then the effect of a growing heading error on the drift in position would be magnified and hence a larger worst case drift would be expected.

To investigate the error characteristics of the PDR filter eight more walks were completed, each consisting of 800 steps (excluding the initial alignment phase). The results obtained for the three cases for each of the walks are shown in Figure 4.11. For the PDR filter the majority of these examples suffer from heading and vertical displacement errors that grow in one direction over time, however the direction of these errors and the rate at which they grow varies from one example to the next. Figure 4.12 shows that the drift in heading behaves as a random walk about a linearly growing value in most cases, with a worst case final error of $16.8°$. This can be explained by uncorrected bias errors in the angular velocity measurements obtained from the IMU. Although the biases are estimated when the IMU is switched on, it is not possible to determine them exactly. Furthermore bias instability effects cause the biases to change over time. This may explain case (e), in which the direction in which heading errors were accumulating changed after around 310 step events. Figure 4.13 shows the drift in the (x, y) plane during each run. Note that the drift is highly dependent on the error in

heading. The walks during which only small heading errors were incurred also had relatively small worst case drifts in horizontal position (e.g. walks (d) and (g)), whereas large heading errors caused large drifts in horizontal position at either end of the corridor (e.g. walks (b) and (c)). Hence, at least in this scenario, heading errors are the primary cause of the worst case errors in horizontal position. The worst case drift in horizontal position recorded in any of the walks was $6.42\,\mathrm{m}$. Figure 4.14 plots the drift in vertical displacement for each walk. As with heading errors, the error in vertical displacement increases about a linearly growing value, however the magnitude and direction of this value is again different for each run. The worst case error in vertical displacement was $5.47\,\mathrm{m}$. Note that the worst case errors in both horizontal and vertical drift are small relative to the total distances travelled by the user, which were approximately $500\,\mathrm{m}$ for each of the examples.

Although it is clear that there are systematic errors in the step events generated by the PDR filter, it is not clear how they can be estimated (and hence removed) without the benefit of ground truth data. It is likely that systematic errors in the measurements received from the IMU are partly to blame. Such errors can only be removed if they can be precisely measured, which is not possible in this case. Systematic errors may also be introduced due to the mounting point of the IMU on the user's foot. For example the IMU may be mounted at a position where it moves very slightly during each stance phase. If such a movement occurs in the same direction during each stance phase then a systematic error will result. To investigate such error sources in more detail would require a ground truth for the acceleration and angular velocity of the IMU, which is not available from current technology.

Figure 4.15 shows the path calculated by the PDR filter over a larger area of the William Gates building. The initial position and orientation were obtained using an initialisation phase in which range measurements from the Active Bat system were used. The Active Bat system is installed along the corridor in the upper left corner of Figure 4.15(a) where the path begins. This walk demonstrates the ability of the PDR filter to measure changes in vertical displacement, such as when a user ascends or descends a flight of stairs. By the end of the path a small error in the user's heading had accumulated (visible at the top right of Figure 4.15(b)), causing a final error of $9.31\,\mathrm{m}$ in the (x, y) plane (calculated using the final position obtained from the Active Bat system as ground truth). The error in the final vertical displacement was $-0.48\,\mathrm{m}$. This example will be revisited throughout Chapters 5 and 6.

4.3 Use of magnetometers

One way to prevent drift in heading is to make use of the magnetometers that are contained within IMUs such as the Xsens Mtx. The magnetometers measure magnetic field strength in three dimensions at the same sampling frequency as the gyroscopes and accelerometers (a

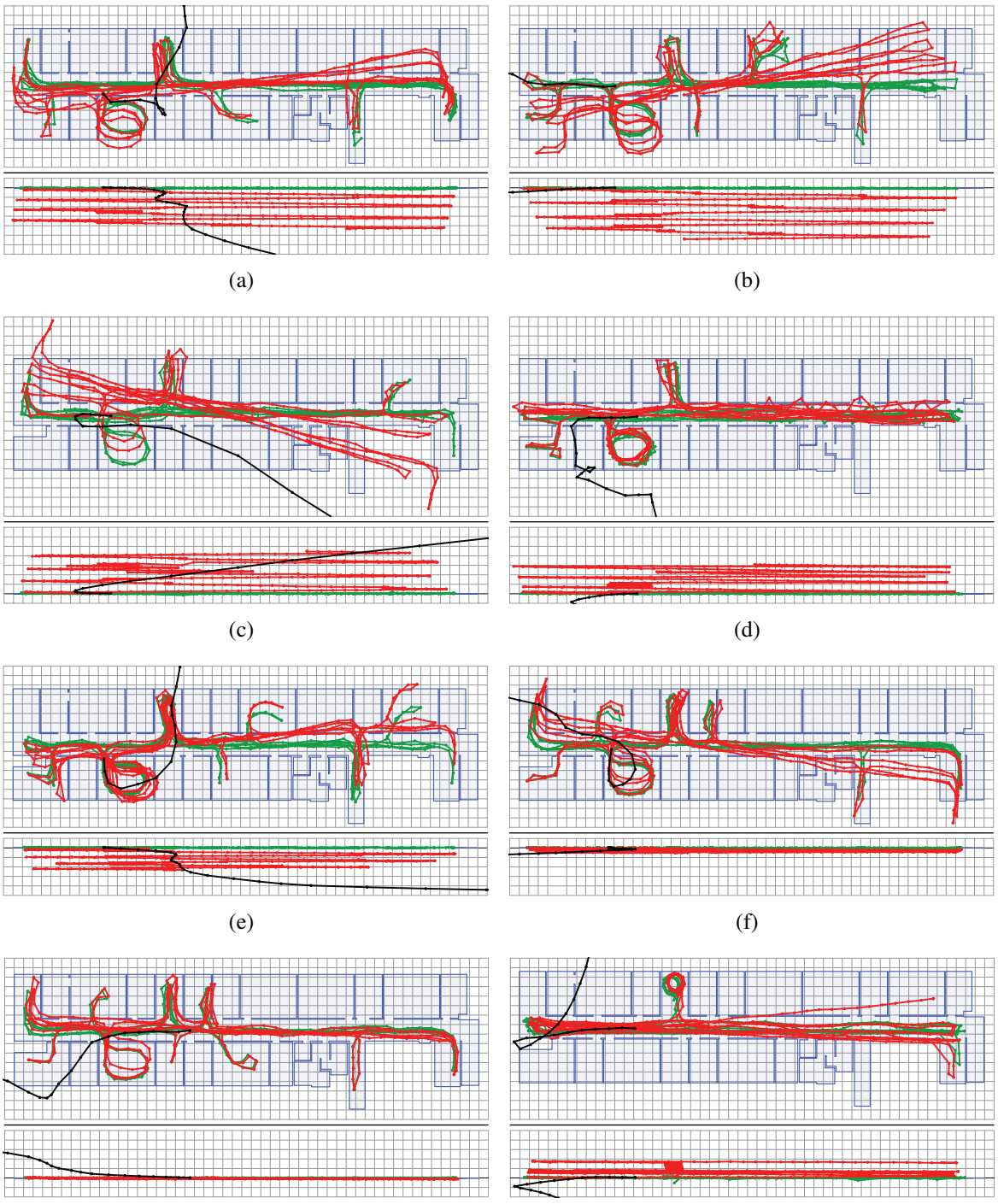


Figure 4.11: Eight more examples of the paths generated by the unconstrained INS (black), the PDR fitler (red), and the ground truth filter (green). Each example consists of 400 step events. Grid size $1\,\mathrm{m}^2$.

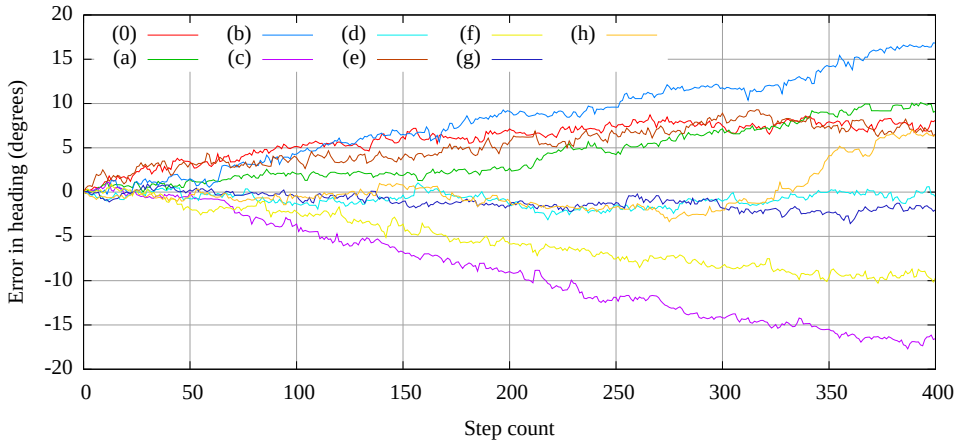


Figure 4.12: The error in heading incurred by the PDR filter during nine walks. Walks (a-h) are shown in Figure 4.11. Walk (0) is shown in Figure 4.9.

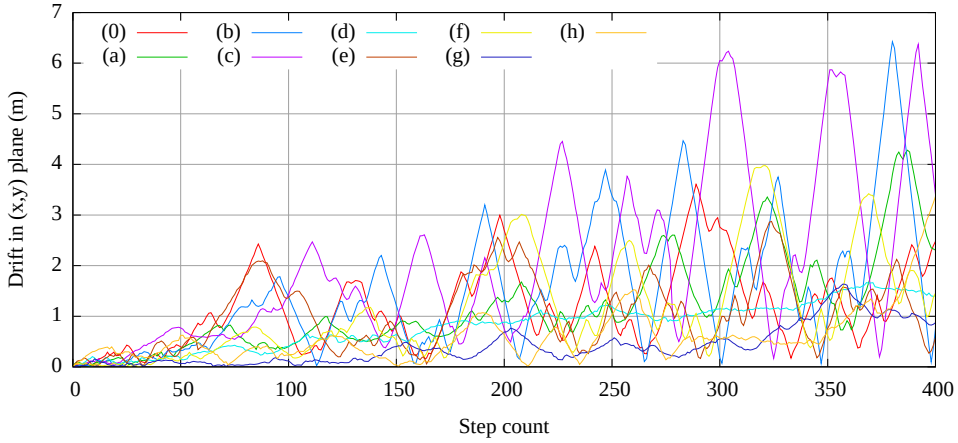


Figure 4.13: The drift in horizontal position incurred by the PDR filter during nine walks. Walks (a-h) are shown in Figure 4.11. Walk (0) is shown in Figure 4.9.

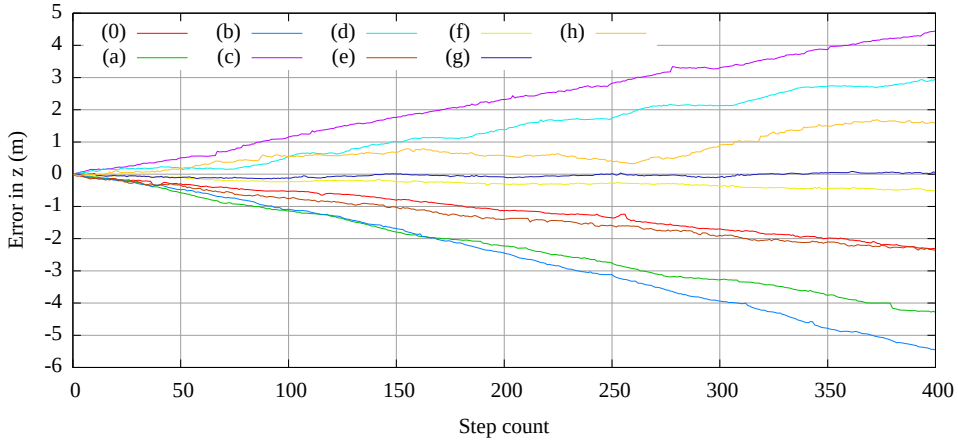


Figure 4.14: The error in vertical displacement incurred by the PDR filter during nine walks. Walks (a-h) are shown in Figure 4.11. Walk (0) is shown in Figure 4.9.

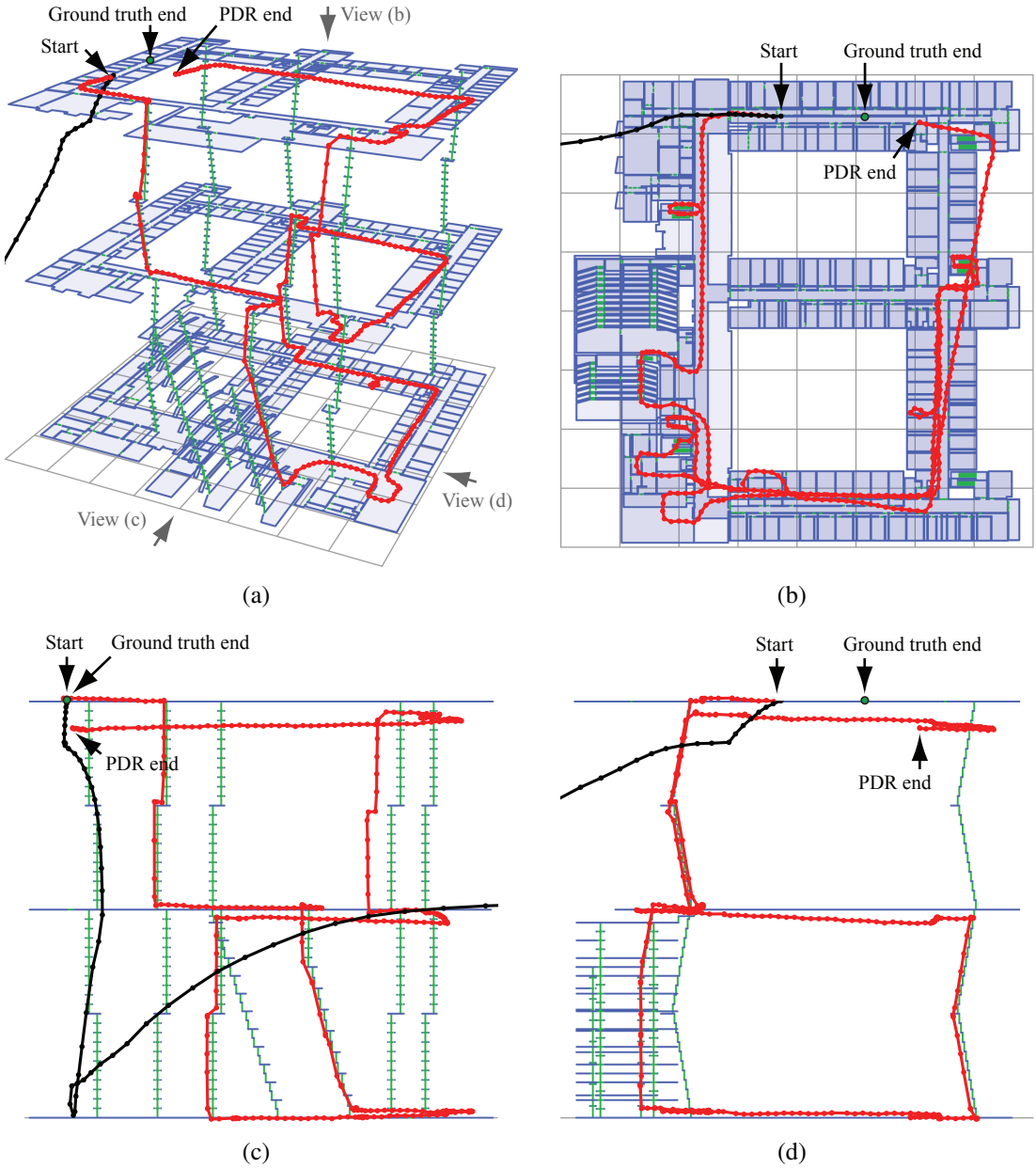


Figure 4.15: A path of step events calculated by the PDR filter as a user walked throughout the William Gates building (red). The path calculated by unconstrained inertial navigation is also shown (black). Figure (a) is annotated with the view points of Figures (b-d). Grid size $10\,\mathrm{m}^2$. Diagrams are exploded 10x in the z-axis.

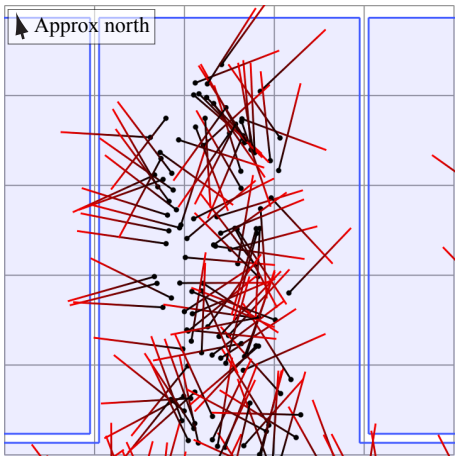


Figure 4.16: Magnetic field vectors measured in a single room by the IMU and projected into the global frame of reference. A black dot indicates the point from which each vector was measured.

sampling frequency of $100\,\mathrm{Hz}$ is used in this thesis). In the absence of any magnetic disturbances the magnetic field strength vectors obtained from the magnetometers should always point towards the magnetic north pole, making it possible to determine the absolute heading of the IMU.

Unfortunately the approach described above does not work well in the William Gates building, where there are many strong local magnetic disturbances. In particular metallic floor panels are installed throughout much of the building, causing local disturbances that are far stronger than the earth's magnetic field. The close proximity of the foot-mounted IMU to these panels prevents the magnetometer from determining the absolute heading of the device with any degree of reliability. This is illustrated by Figure 4.16, in which normalised magnetic field strength vectors obtained during an example walk are plotted in the global frame of reference (position and orientation estimates obtained from the ground truth filter described in Section 4.2.2 were used to position each vector and project them correctly into the global frame of reference). In the absence of any magnetic disturbances all of the vectors should point in the same direction towards the magnetic north pole, however this is clearly not the case for the measurements obtained.

It is possible to develop advanced Kalman filters that attempt to detect and adapt to magnetic disturbances [81], however in this case such techniques are of limited use because strong magnetic disturbances are nearly always present. Hence this thesis does not consider the use of magnetometers in this way. Instead algorithms that limit heading drift based on the use of environmental constraints are presented in Chapter 5. An alternative way in which data from the IMU's magnetometer can be used is outlined in Section 7.2.

4.4 Conclusions

This chapter has developed and evaluated a PDR filter suitable for providing relative movement information to a higher level localisation filter. The requirements of such a filter differ from those a standalone PDR system, requiring both the segmentation of the user's movement into coarse step events and the application of additional position assertions to in order to prevent the filter from compensating for errors in previously generated events.

It is important to remember that the PDR filter described in this chapter is only able to track the *relative* movements of a pedestrian. Both systematic and random errors during each run cause drift in vertical displacement, as well as drift in heading that in turn causes the accumulation of errors in the calculated position over time. When using an Xsens Mtx IMU the filter incurred worst case drifts of $6.42\,\mathrm{m}$ horizontally and $5.47\,\mathrm{m}$ vertically throughout a number of test walks, each of which was around $500\,\mathrm{m}$ in length. Hence although the PDR filter may be sufficient for tracking a user for one or two minutes, it cannot provide metre-level accuracy (as is required by many indoor location-aware applications) for longer periods of time.

Although systematic errors are responsible for the majority of the drift during each run, it is not known how the magnitude and direction of such errors can be determined in advance. Heading errors can in theory be corrected using magnetometers, but in practice it is not always possible to do so due to strong magnetic disturbances in buildings. This thesis assumes that such errors will always be present and instead develops algorithms that make use of environmental constraints to correct them in a higher level localisation filter, which is described in the following chapter.

Chapter 5

Enforcing environmental constraints with particle filters

Pedestrian dead reckoning (PDR) techniques such as the one described in the previous chapter can be used to track the position of a user relative to their initial location and heading. The positional accuracy of such a system degrades over time as errors in the measured sensor data propagate through whichever algorithm is used, accumulating as drift in the calculated position. Furthermore, the accuracy of absolute positions calculated by a PDR system is dependent on the accuracy of the initial location and heading, which must be provided by some other means.

One way to limit the drift incurred by a PDR system is to exploit knowledge of environmental constraints such as walls and floors, which naturally limit the movement of a pedestrian in an indoor environment. This chapter shows that a map of such constraints can be used not only to limit drift, but also to automatically infer the user's absolute position by finding a unique path through the environment that is consistent with the relative movements reported by the

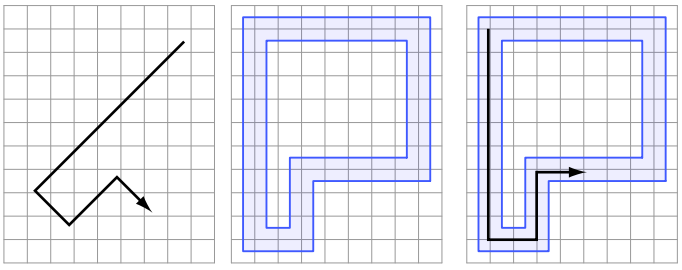


Figure 5.1: The pedestrian localisation problem: To determine the absolute location of a pedestrian given (a) a path of step events describing their relative movement, and (b) a map of the environment. The solution is shown in (c), in which the pedestrian's absolute position has been determined.

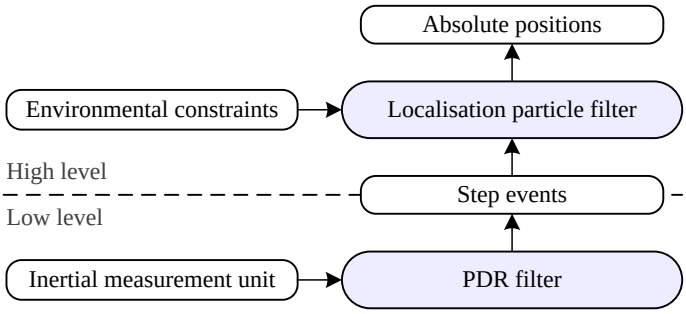


Figure 5.2: The structure of a pedestrian localisation and tracking system.

PDR system. The task of automatically inferring the user's position is known as the pedestrian localisation problem and is illustrated in Figure 5.1.

This chapter describes a solution to the pedestrian localisation problem in which a particle filter is used to locate and subsequently track the user in an indoor environment. Step events generated by a PDR filter are used as inputs to the filter, as shown in Figure 5.2. In theory any PDR filter could be used, provided that the error in each step event does not grow larger over time (i.e. the error in the 10^{th} step event will be in the same order of magnitude as the error in the 1000^{th} step event). This is the case for the PDR filter developed in Chapter 4 due to the application of zero velocity updates and position assertions to effectively reset the filter during each stance phase.

In this chapter the PDR filter is considered as a black box, reporting step events of the form

$$\boldsymbol{e}_t = (l, \delta z, \delta\theta, \xi) \tag{5.1}$$

as described in Section 4.1.2. Each step event has a corresponding error model describing the measurement's uncertainty. This model is specified by a distribution $\mathcal{H}_{L,\Theta}$ of the error in $(l, \delta\theta)$ and $\mathcal{V}_Z$ of the error in δz. These distributions are specified separately due to the different way in which individual components of the step event are used by the filter (random variables are sampled from $\mathcal{H}_{L,\Theta}$, whereas probability densities are read directly from $\mathcal{V}_Z$), as described in Section 5.2. The notation

$$(l_e, \delta\theta_e) \backsim \mathcal{H}_{L,\Theta} \tag{5.2}$$

will be used to indicate that $(l_e, \delta\theta_e)$ is sampled from the distribution $\mathcal{H}_{L,\Theta}$. The notation

$$p = \mathcal{V}_Z(\delta z_e) \tag{5.3}$$

indicates that p is the probability that the error in the measurement δz is δz_e. In theory the error model corresponding to each step event could be generated by the PDR filter itself, however in this chapter a generic error model is assumed for each step, as described in Section 5.3.

The remainder of the chapter is set out as follows. Section 5.1 introduces a data-structure that is suitable for representing buildings with multiple floors. Section 5.2 describes the design of a particle filter to solve the pedestrian localisation problem. The filter's localisation and tracking functionalities are described in Sections 5.3 and 5.4 respectively.

5.1 Representing building constraints

In order to use environmental constraints to help locate and track a pedestrian's position it is necessary to represent them in a machine-readable form. In previous work on robot localisation 2-dimensional maps have been used [57, 66]. Supporting environments with multiple floors was not necessary since the robots used were not able to ascend or descend stairs. Such robots typically used laser range-finders to scan for obstructions in a single horizontal plane for which a 2-dimensional representation of the environment was sufficient.

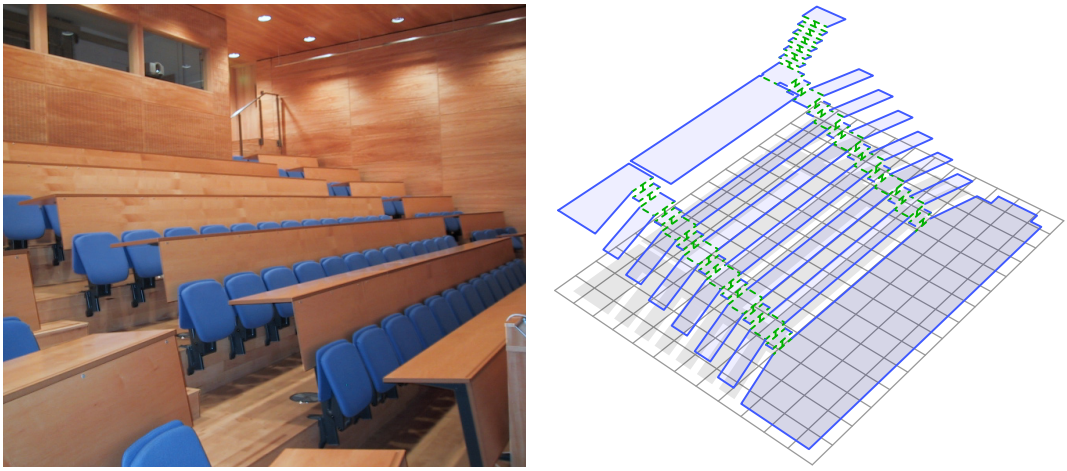
In the pedestrian localisation problem the primary type of 'measurement' is to use the movement of a pedestrian to detect the *absence* of an obstruction. Since a pedestrian is constrained by gravity to be in contact with the floor, obstructions on a single floor of a building can be represented by a 2-dimensional map. Unfortunately pedestrians frequently ascend and descend stairs and so a solely 2-dimensional representation is not sufficient. A more suitable representation is a 2.5-dimensional map, as described below.

A 2.5-dimensional map is defined as a collection of rooms, each consisting of one or more planar floor polygons. Each floor polygon corresponds to a surface in the building on which a pedestrian's feet may be grounded. Each edge of a floor polygon is either the edge of an impassable obstruction or a connection to the edge of another polygon. A connection represents an edge through which a pedestrian can pass, such as a doorway. Each connection is composed of two edges, one associated with each of the connected polygons. These edges must coexist in the horizontal (x, y) plane, however they may be separated in the vertical (z) direction to allow the representation of stairs. It is possible to represent even complex rooms using this format, such as the lecture theatre shown in Figure 5.3.

Figure 5.4 shows how a 2.5-dimensional map data-structure is stored in memory. Each floor polygon is defined by a packed array of vertices

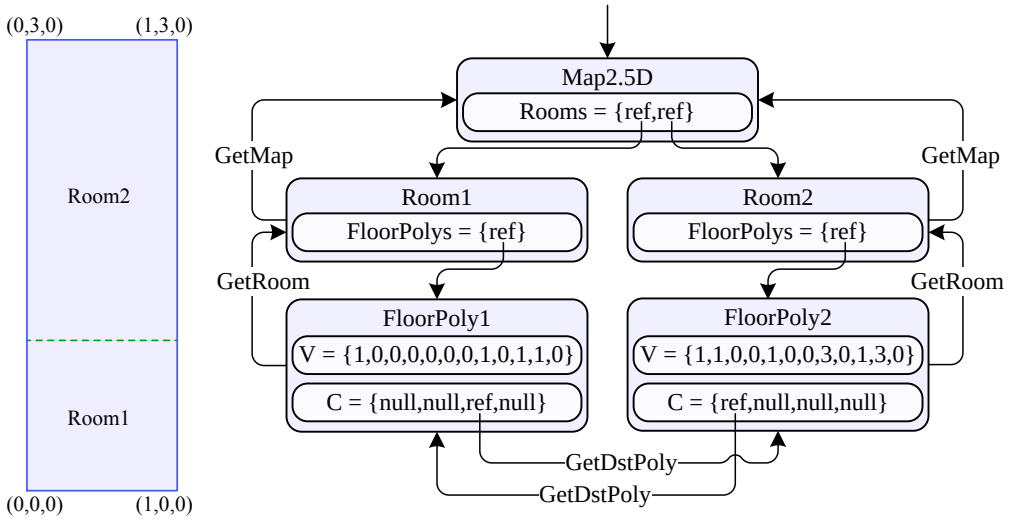
$$\boldsymbol{V} = (x_1, y_1, z_1, x_2, y_2, z_2, ..., x_n, y_n, z_n) \tag{5.4}$$

which are specified in clockwise order. This vector representation avoids having to approximate the map to a specified granularity and allows large maps to be represented efficiently in



(a) (b)

Figure 5.3: (a) A lecture theatre. (b) Its 2.5-dimensional representation. The edges separating adjacent rows of seating are treated as walls (solid blue lines), whereas the edges between stairs in the aisle are connections (dashed green lines). Grid size = 1 m^2.



(a) (b)

Figure 5.4: (a) A simple 2.5-dimensional map. (b) Its representation in memory. Arrows correspond to references. Where applicable, a label indicates the name of the method which can be used to obtain the referred object.

memory. The edges of a floor polygon are given by

$$\boldsymbol{E_1} = \overrightarrow{(x_1, y_1, z_1)(x_2, y_2, z_2)} \tag{5.5}$$

$$\boldsymbol{E_2} = \overrightarrow{(x_2, y_2, z_2)(x_3, y_3, z_3)} \tag{5.6}$$

$$\ldots$$

$$\boldsymbol{E_n} = \overrightarrow{(x_n, y_n, z_n)(x_1, y_1, z_1)}\,. \tag{5.7}$$

Connections are defined by an array of references to other polygons $\boldsymbol{C}$, where $\boldsymbol{C}[i]$ is a reference to the polygon to which edge $\boldsymbol{E_i}$ connects, or null if that edge is a wall. Note that a connection between two floor polygons A and B is represented by two references, one from A to B and another from B to A. Hence in theory it is possible to represent a uni-directional connection by having an unmatched reference from one polygon to another. This may be useful in some cases, for example to represent a one-way ticket barrier.

Some of the functions that can be performed on 2.5-dimensional maps are listed below and are used in the remainder of this chapter.

- $\mathrm{GetDstPoly}(\boldsymbol{E})$ returns the polygon to which the specified edge connects, or null if the edge is a wall:

$$\mathrm{GetDstPoly}(\boldsymbol{E}_i) = \boldsymbol{C}[i] \tag{5.8}$$

- $\mathrm{Type}(\boldsymbol{E})$ returns the type of edge $\boldsymbol{E}$:

$$\mathrm{Type}(\boldsymbol{E}) = \begin{cases} \mathrm{Connection} & \text{if } \mathrm{GetDstPoly}(\boldsymbol{E}) \neq \mathrm{null} \\ \mathrm{Wall} & \text{otherwise} \end{cases} \tag{5.9}$$

- $\mathrm{GetRoom}(\mathrm{poly})$ returns the room to which the specified floor polygon belongs.

- $\mathrm{Height}(x, y, \mathrm{poly})$ returns the height of the specified polygon at position (x, y) in the horizontal plane. Usually the height is constant across the polygon's surface (since most floor surfaces in buildings are flat), in which case (x, y) is ignored and the constant height of the polygon is returned. Note however that sloping floor polygons are permitted (the 2.5-dimensional map data-structure only requires floor polygons to be planar) and can be used to represent sloping surfaces in the environment. A sloping floor polygon can also be used to represent a staircase rather than having to define each step individually.

5.2 Localisation filter design

Recall from Section 2.3.5 that particle filters approximate the belief $\mathrm{Bel}(\boldsymbol{x}_t)$ of the state at time t as a set of weighted samples known as *particles*

$$\mathcal{S}_t = \{\langle \boldsymbol{x}_t^i, w_t^i \rangle \;\; i = 1, ..., N\} \tag{5.10}$$

where $\boldsymbol{x}_t^i$ is the state and w_t^i is the weight of the i^{th} particle. A particle filter update consists of generating the set $\mathcal{S}_t$ from the previous posterior $\mathcal{S}_{t-\delta t}$. In this section a pedestrian localisation filter is developed based on the simple Bootstrap particle filter, in which $\mathcal{S}_t$ consisting of N particles is generated from the previous state as follows [55].

1. **Propagation:** For $i = 1, ..., N$ sample the updated state from the motion model distribution:
$$\boldsymbol{x}_t^i \backsim p(\boldsymbol{x}_t | \boldsymbol{x}_{t-\delta t}^i, \boldsymbol{u}_t)\,. \tag{5.11}$$

2. **Correction:** For $i = 1, ..., N$ update the importance weight:
$$w_t^i = w_{t-\delta t}^i \cdot p(\boldsymbol{z}_t | \boldsymbol{x}_t^i) \tag{5.12}$$

3. **Re-sampling:** Create N new states by randomly sampling the propagated states in proportion to their importance weights. $\mathcal{S}_t$ is then the set of N particles with the sampled states and equal weights $w = 1/N$.

In practice the propagation and correction steps are often combined into a single particle update step, since this avoids having to iterate twice over the set of particles.

In the pedestrian localisation filter each particle has a state

$$\boldsymbol{x}_t = (x_t, y_t, z_t, \theta_t, \mathrm{poly}_t) \tag{5.13}$$

at time t, where (x_t, y_t, z_t) is the position, θ_t is the heading and poly_t is a reference to the floor polygon to which the particle is constrained. This reference is crucial to the performance of the filter since it acts as an index into the 2.5-dimensional map data-structure. This reference can be used to discover nearby constraints (i.e. those defined by the edges of poly_t) in $\mathcal{O}(1)$ time[1], with the size of the map having no effect on the lookup time. The position (x_t, y_t, z_t) is constrained to lie on the surface of poly_t. Hence the vertical displacement of the particle is also given by

$$z_t = \mathrm{Height}(x_t, y_t, \mathrm{poly}_t)\,. \tag{5.14}$$

[1] An excellent description of $\mathcal{O}$-notation (pronounced "big oh notation") can be found in [82].

and does not need to be stored explicitly in the state vector. Despite this it is often useful to do so in practice, since it allows the state vector to be passed directly into functions that expect a vector containing the particle's 3-dimensional position. In particular particle state vectors can be passed directly as vertices into the functions of a 3-dimensional rendering library.

The following sections outline in detail the propagation, correction and re-sampling steps of the localisation filter.

5.2.1 Particle propagation

The propagation step generates the updated state $\boldsymbol{x}_t$ of a particle by sampling the motion model distribution $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-\delta t}, \boldsymbol{u}_t)$, where $\boldsymbol{x}_{t-\delta t}$ is the state of the particle at time $t - \delta t$ and $\boldsymbol{u}_t = (l, \delta\theta, \xi)$ is the step event $\boldsymbol{e}_t$ excluding the δz component. By using $\boldsymbol{u}_t$ directly in the propagation step the particle filter is able to propagate particles in a way that is consistent with the measured step length l, change in heading $\delta\theta$ and heading offset ξ (the δz component of the step event is used separately in the correction step for reasons described below).

An alternative approach would be to assume a generic motion model that does not depend on the step event (for example one that assumes that the user is equally likely to have moved up to some maximum distance in any direction) before using all of the components of $\boldsymbol{e}_t$ to weight the resulting particles during the correction step. In this case particles that were propagated by distances and headings consistent with the step event would be assigned large weights during the correction step, whilst other particles would be assigned small weights. A major drawback of this approach is that the majority of the particles would be propagated in a way that is inconsistent with $\boldsymbol{e}_t$. As a result most of the particles would be assigned very small weights during the correction step and so would contribute very little towards the representation of the posterior distribution. It would therefore be necessary to propagate far more particles than are actually needed to represent the posterior distribution.

The approach used in this thesis ensures that all of the particles are propagated in a way that is consistent with the components of the step event contained by $\boldsymbol{u}_t$. Note however that the δz component of the step event *is* still used in the correction step (as described in Section 5.2.2) rather than during propagation. This is because particles are by definition constrained to lie on the surface of the 2.5-dimensional map. Hence the vertical displacement of each particle must be updated according to Equation 5.14, rather than according to the measured change in vertical displacement.

The updated state $\boldsymbol{x}_t$ of a particle is calculated from its previous state $\boldsymbol{x}_{t-\delta t}$ and $\boldsymbol{u}_t$ as follows. First, the horizontal components of the step event are perturbed according to the step's

uncertainty model

$$l' = l + l_e \tag{5.15}$$
$$\delta\theta' = \delta\theta + \delta\theta_e \tag{5.16}$$

where l_e and $\delta\theta_e$ are drawn from the step uncertainty model

$$(l_e, \delta\theta_e) \backsim \mathcal{H}_{L,\Theta}\,. \tag{5.17}$$

The new heading and (x, y) position are then calculated from the previous state and the perturbed step parameters

$$\theta_t = \theta_{t-\delta t} + \delta\theta' \tag{5.18}$$
$$x_t = x_{t-\delta t} + l' \cdot \cos(\theta_t - \xi) \tag{5.19}$$
$$y_t = y_{t-\delta t} + l' \cdot \sin(\theta_t - \xi) \tag{5.20}$$

where ξ is the offset between the user's heading and the step direction as specified in the step event.

It is also necessary to determine the floor polygon poly_t to which the propagated particle is constrained. Initially it is assumed that the floor polygon in which the particle resides is unchanged by the update

$$\mathrm{poly}_t = \mathrm{poly}_{t-\delta t}\,. \tag{5.21}$$

In order for the particle to have exited this floor polygon, the step vector must intersect one of its edges in the (x, y) plane. Let $\boldsymbol{A} = (x_{t-\delta t}, y_{t-\delta t})$ and $\boldsymbol{B} = (x_t, y_t)$ be the start and end points of the step vector in the (x, y) plane. Secondly, let

$$\mathrm{Intersect}(\boldsymbol{A}, \boldsymbol{B}, \mathrm{poly}_t) = [\boldsymbol{C}, \boldsymbol{E}] \tag{5.22}$$

be a function that calculates the first intersection between the step vector $\overrightarrow{\boldsymbol{AB}}$ and the edges of poly_t, returning both the point of intersection $\boldsymbol{C}$ and the edge $\boldsymbol{E}$ on which the intersection occurs. This function is used to update poly_t according to one of three cases:

1. $[\boldsymbol{C}, \boldsymbol{E}] = [\mathrm{null}, \mathrm{null}]$: No intersection point is found. The particle must still be constrained to the same polygon, so no update is necessary.

2. $\mathrm{Type}(\boldsymbol{E}) = \mathrm{Wall}$: The first intersection is with a wall. In this case the particle's weight will be set to zero in the correction step, enforcing the constraint that walls are impassable. Since this particle will never be re-sampled, it is not necessary to update poly_t.

3. $\mathrm{Type}(\boldsymbol{E}) = \mathrm{Connection}$: The first intersection is with an edge connecting to another polygon. In this case the polygon must be updated

$$\mathrm{poly}_t = \mathrm{GetDstPoly}(\boldsymbol{E})\,. \tag{5.23}$$

Since a single step may span multiple floor polygons, the intersection test is repeated between the remainder of the step vector $\overrightarrow{\boldsymbol{CB}}$ and the updated polygon. This process continues recursively until one of the first two cases applies.

Finally, if the vertical position z_t is to be stored explicitly in the state vector then it is calculated using Equation 5.14.

5.2.2 Particle correction

The correction step updates the weight w_t of a propagated particle. In the localisation filter this is used to enforce environmental constraints. If a wall is intersected by the particle during the propagation step then it is assigned a weight

$$w_t = w_{t-\delta t} \cdot 0 = 0\,. \tag{5.24}$$

If a wall is not intersected, the particle's weight is updated based on how closely the change in height δz of the current step event matches the change in height of the particle

$$\delta z_{\mathrm{poly}} = z_t - z_{t-\delta t}\,. \tag{5.25}$$

In this case the particle's weight is calculated as

$$w_t = w_{t-\delta t} \cdot \mathcal{V}_Z(|\delta z - \delta z_{poly}|) \tag{5.26}$$

where $\mathcal{V}_Z$ is part of the step event's error model (see page 88). Note that the change in vertical displacement reported in the step is used as a measurement in the Bayesian framework. Particles whose change in height over the step closely matches the change in height reported in the step event are assigned larger weights.

The combined particle propagation and correction algorithm is given by Algorithm 2. The algorithm executes in $\mathcal{O}(1)$ time in practice and hence can be applied to all N particles in $\mathcal{O}(N)$ time. Note that the cost of applying Algorithm 2 to a single particle does not depend on the size of the environment. This is due to the use of poly_t as an index from the particle into the corresponding part of the map data-structure. This is crucial for deployments in very large environments. One part of the algorithm for which $\mathcal{O}(1)$ complexity is not obvious is the while loop that updates the floor polygon to which the particle is constrained (lines 10 -

21). The number of loop iterations is equal to the number of polygons through which the step update passes. This value is bounded by a combination of the user's maximum step length and the fact that in practice floor surfaces in buildings have a minimum size. Within each iteration the cost of the $\mathrm{Intersect}$ function is dependent on the number of edges of the current floor polygon, however this value is also bounded in practice since floor surfaces tend to have a maximum number of edges (i.e. most rooms have four). Hence updating the floor polygon consists of executing an $\mathcal{O}(1)$ function $\mathcal{O}(1)$ times and hence is itself $\mathcal{O}(1)$.

5.2.3 Re-sampling

Consider a filter where $\mathcal{S}_t$ is simply the set of propagated and corrected particles. Over time the number of particles with non-zero weights will decrease as more and more particles violate wall constraints. More generally the distribution of particle weights becomes skewed, with fewer and fewer particles having non-negligible weights. This happens not only as a result of wall intersections, but also when the user descends or ascends a flight of stairs. In this case any particles that are not constrained to stair regions will have weights close to zero after a few updates. Hence only a very small proportion of the N particles contribute to the representation of $\mathrm{Bel}(\boldsymbol{x}_t)$ and as a result the distribution is not adequately represented. Furthermore computation is wasted updating particles that have negligible or zero weights. This is known as the *weight degeneracy* problem.

Re-sampling solves the weight degeneracy problem by generating a new set of particles from the updated (i.e. propagated and corrected) set whose weights are less skewed. Many re-sampling schemes have been proposed [83]. The localisation filter described in this chapter implements the simplest approach, known as multinomial re-sampling. Multinomial re-sampling generates a new set of N particles with equal weights $w = 1/N$. The state of each new particle is randomly sampled from the states of the updated particles in proportion to their weights. Thus states with high weights tend to be replicated whilst states with low weights tend to be removed.

Multinomial resampling is implemented as follows. First the accumulated weights

$$\mathrm{Acc} = (\mathrm{acc}_t^0, ..., \mathrm{acc}_t^N) \tag{5.27}$$

are calculated in $\mathcal{O}(N)$ time, where

$$\mathrm{acc}_t^i = \sum_{n=1}^{i} w_t^n \,. \tag{5.28}$$

Algorithm 2 Update - Applies the propagation and correction algorithms to generate a particle at time t from a particle at time $(t - \delta t)$.

procedure UPDATE($\{(x_{t-\delta t}, y_{t-\delta t}, z_{t-\delta t}, \theta_{t-\delta t}, \text{poly}_{t-\delta t}), w_{t-\delta t}\}$, $\boldsymbol{e}_t = (l, \delta z, \delta\theta, \xi)$)
    // Perturb the step event
    $(l_e, \delta\theta_e) \backsim \mathcal{H}_{L,\Theta}$
    $l' \leftarrow l + l_e$
    $\delta\theta' \leftarrow \delta\theta + \delta\theta_e$
    // Update the particle's xy-position
    $\theta_t \leftarrow \theta_{t-\delta t} + \delta\theta'$
    $x_t \leftarrow x_{t-\delta t} + l' \cdot \cos(\theta_t - \xi)$
    $y_t \leftarrow y_{t-\delta t} + l' \cdot \sin(\theta_t - \xi)$
    // Initialise the intersection algorithm
    $\text{poly}_t \leftarrow \text{poly}_{t-\delta t}$, $\boldsymbol{A} \leftarrow (x_{t-\delta t}, y_{t-\delta t})$, $\boldsymbol{B} \leftarrow (x_t, y_t)$
    done $\leftarrow$ false , kill $\leftarrow$ false
    // Determine the polygon to which the updated particle is constrained
    while ¬done **do**
        $[\boldsymbol{C}, \boldsymbol{E}] \leftarrow \text{Intersect}(\boldsymbol{A}, \boldsymbol{B}, \text{poly}_t)$
        if $\boldsymbol{C} = \text{null}$ **then**
            done $\leftarrow$ true
        else if $\text{Type}(\boldsymbol{E}) = \text{Wall}$ **then**
            // Note the wall intersection
            kill $\leftarrow$ true
            done $\leftarrow$ true
        else if $\text{Type}(\boldsymbol{E}) = \text{Connection}$ **then**
            $\text{poly}_t \leftarrow \text{GetDstPoly}(\boldsymbol{E})$
            $\boldsymbol{A} \leftarrow \boldsymbol{C}$
        end if
    end while
    // Update the particle's z-position
    $z_t \leftarrow \text{Height}(x_t, y_t, \text{poly}_t)$
    // Re-weight the particle
    if kill **then**
        $w_t \leftarrow 0$
    else
        $\delta z_{\text{poly}} \leftarrow z_t - z_{t-\delta t}$
        $w_t \leftarrow w_{t-\delta t} \cdot \mathcal{V}_Z(|\delta z - \delta z_{poly}|)$
    end if
    // Return the updated particle
    return $\langle (x_t, y_t, z_t, \theta_t, \text{poly}_t), w_t \rangle$
end procedure

The state of each new particle is then sampled by drawing a random value

$$r \sim \mathcal{U}(0, \mathrm{acc}_t^N] \tag{5.29}$$

where $\mathcal{U}(0, \mathrm{acc}_t^N]$ is the uniform distribution from 0 (inclusive) to acc_t^N (exclusive). The sampled state is then the state of the j^{th} particle, where j is the smallest value such that $r < \mathrm{acc}_t^j$. This value is found by a binary search for r in Acc which requires $\mathcal{O}(\log_2 N)$ time. Since this is repeated for each of the N new particles, the overall complexity of the re-sampling step is $\mathcal{O}(N \log_2 N)$.

Note that it is impossible for the state of a particle with a weight of zero to be sampled by the algorithm above. Hence the states of any particles found to violate wall constraints are removed during the subsequent re-sampling step. Note also that when the user descends or ascends a flight of stairs the particle cloud will quickly converge to stair regions. This will occur because particles constrained to such regions will be assigned large weights in the correction step relative to other particles, as described in Section 5.2.3. Hence their states are more likely to be replicated during re-sampling.

5.3 Initialisation and localisation

Figure 5.5 shows an example in which the localisation particle filter is applied in the William Gates building. Figure 5.5(a) shows the path of step events generated by the PDR filter that was developed in Chapter 4. Note that this path is the first part of the one shown in Figure 4.15 (page 84), which is used as a running example throughout this thesis. It has been aligned with the map to have the correct initial location and orientation (using measurements provided by the Active Bat system), both of which are unknown to the localisation algorithm.

A simplified step uncertainty model was used, which was assumed to be constant for all steps. The error of each component in a step event was assumed to be independent, Gaussian distributed and have a mean of zero[2]. Hence the uncertainty model was given by

$$\mathcal{H}_{L,\Theta}(l_e, \delta\theta_e) = \mathcal{N}_{0,\sigma_L}(l_e) \cdot \mathcal{N}_{0,\sigma_\Theta}(\delta\theta_e) \tag{5.30}$$

$$\mathcal{V}_Z(\delta z_e) = \mathcal{N}_{0,\sigma_Z}(\delta z_e) \tag{5.31}$$

where $\mathcal{N}_{\mu,\sigma}$ is a Gaussian distribution with mean μ and standard deviation σ. The deviations used were $\sigma_L = 0.12\,\mathrm{m}$, $\sigma_\Theta = 0.4°$ and $\sigma_Z = 0.05\,\mathrm{m}$. These deviations were chosen to deliberately overestimate the magnitude of errors that are present in the step events reported by the

[2]Note that in reality these assumptions are extremely unlikely to hold. The resulting uncertainty model is crude, however works well in practice.

PDR filter, however this overestimation ensures that the localisation filter is able to operate reliably even when unusually large errors are encountered. In contrast an underestimation could result in a failure where all of the particles are found to violate environmental constraints. Given the availability of a highly accurate ground truth positioning system it would have been possible to measure the errors in individual step events generated by the PDR filter and hence derive a suitable uncertainty model directly from experimental data. Unfortunately the ground truth tracking system used in Chapter 4 was not accurate enough for this purpose. Although it was sufficiently accurate to measure the *accumulation* of drift errors over time, it was not accurate enough to measure errors in *individual* step events.

Figure 5.5(b) shows the initial collection of particles sampled from the prior $\mathrm{Bel}(\boldsymbol{x}_0)$, each drawn as a red dot. Since the initial location of the pedestrian is unknown, a uniform distribution over the entire floor surface is used. The initial heading is also unknown, hence the particle headings are distributed uniformly in all directions. In this example $N = 500,000$ particles were used (although in general more particles are required to ensure reliable localisation, as discussed in Section 5.3.2). Figure 5.5(c) shows the particles after the pedestrian had taken eight steps. Figures 5.5(d) and 5.5(e) show the distribution just before and just after the pedestrian started to descend a flight of stairs. Note how the pedestrian is quickly localised to stair regions due to the use of the measured change in vertical displacement in the correction step of the particle filter. Figure 5.5(f) shows the final set of particles, which form a single cluster around the user's true position.

The example described above demonstrates that it is possible for the localisation filter to successfully combine building constraints and relative movement measurements in order to determine the absolute position of a user. It is important to note that the user must be moving in order for his or her position to be determined in this way. Even when the user is moving there are two problems that affect the filter's performance during localisation: symmetry and scalability.

5.3.1 Environmental Symmetry

Figure 5.6 highlights one of the problems faced by the localisation filter. If the layout of a building exhibits translational or rotational symmetry then the relative path described by a sequence of step events is often consistent with multiple different locations. In such cases a cluster of particles forms at each possible location, as can be observed in Figure 5.5(d) and more clearly in Figure 5.6(a). Hence symmetry of the environment can delay or prevent convergence to a single cluster of particles.

For the localisation in Figure 5.5 the user passes through several asymmetries in the environment that allow him to be successfully located. The asymmetrical positioning of the building's

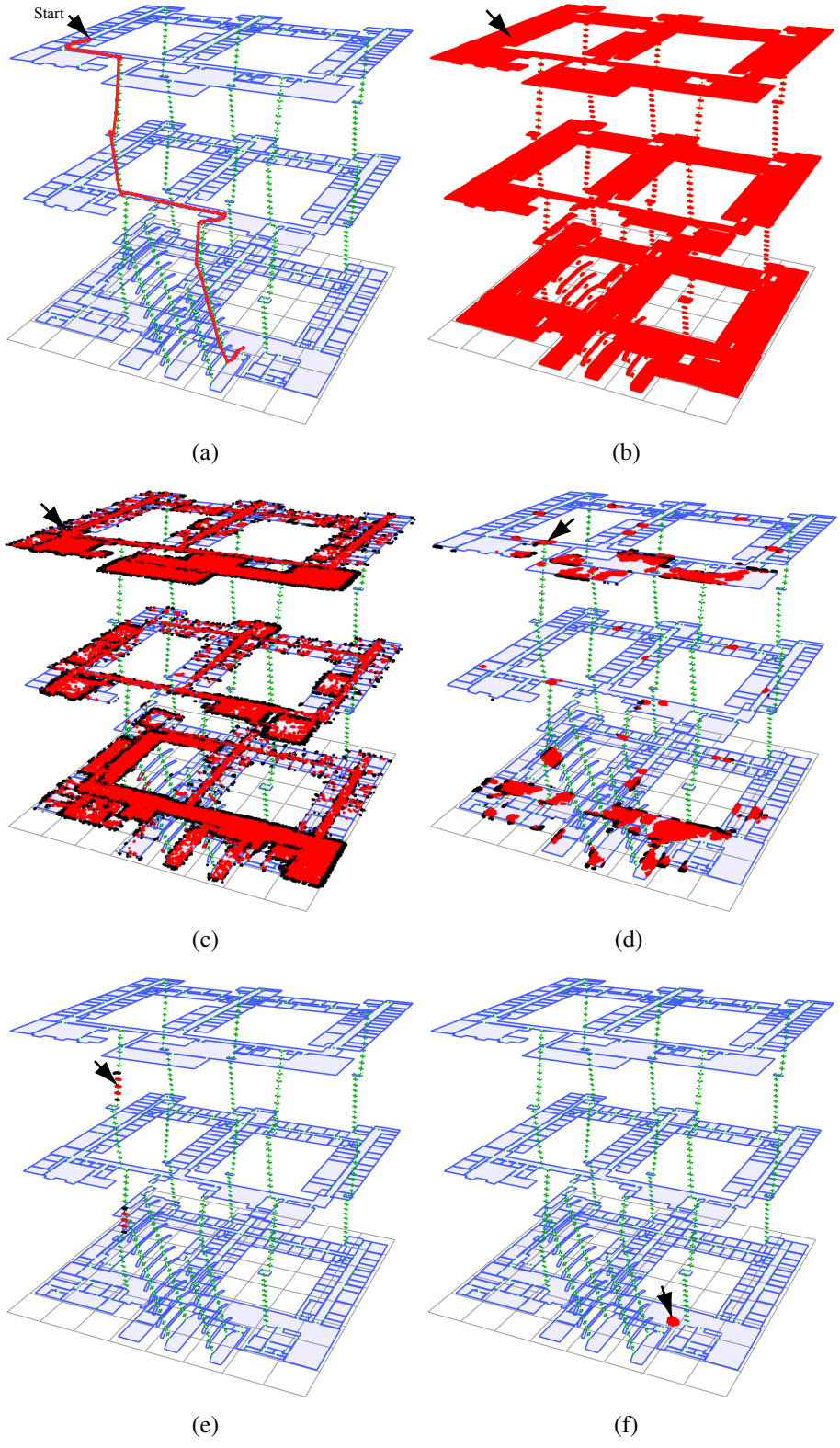
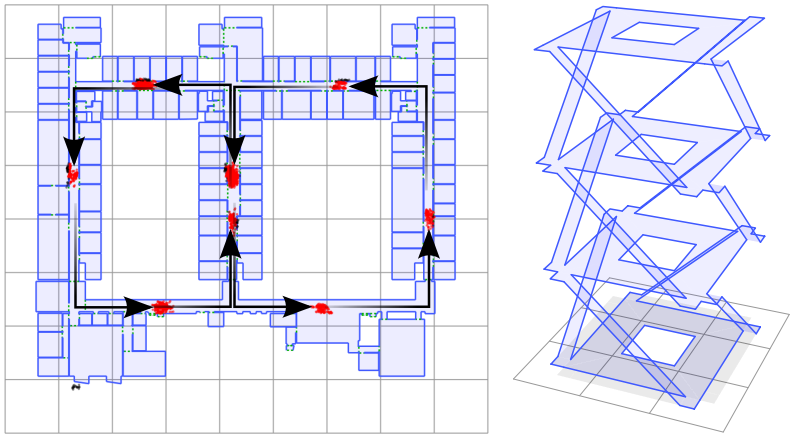


Figure 5.5: An example of localisation in a three storey building. (a) A manually aligned overlay of the path of step events generated by the PDR filter; (b) The prior distribution of $N = 500,000$ particles; (c-f) The particle distribution at four points during localisation. Particles for which $w = 0$ are coloured black. An arrow indicates the actual position of the pedestrian in each figure. Grid size $10\,\mathrm{m}^2$. Diagrams are exploded 10x in the z-axis.



(a) (b)

Figure 5.6: The environmental symmetry problem; (a) Multiple clusters formed due to symmetry on one floor of the William Gates building. The user has walked in a straight line before making a 90° turn to the left. This movement is consistent with eight different locations which arise due to both translational and rotational symmetry. Each arrow indicates the path taken by a cluster to reach its current position. Grid size $= 10\,\mathrm{m}^2$. (b) A hypothetical building for which symmetry prevents the user's position from being determined.

staircases relative to its corridors restricts the user to two possible locations as shown in Figure 5.5(e). The user then passes through a lecture theatre, allowing his position to be uniquely determined.

Since the filter relies on the user passing through asymmetries, it is clear that it is not always possible to determine the user's position. This may be because the user has not yet passed through a sufficient number of asymmetries, or worse, because the environment is not sufficiently asymmetric to make localisation possible. Figure 5.6(b) shows an example of such an environment. Each floor is identical, meaning that the environment exhibits vertical translational symmetry. This makes it impossible to determine the floor on which the user is located unless they visit each one of them (in which case the limits imposed by the bottom and top floors allow the correct floor to be determined). Even if the user does this, there will still be four positions on that floor that could equally correspond to their true position. These arise due to the rotational symmetry exhibited by the environment. Although such an extreme example is unlikely, many buildings do exhibit high degrees of symmetry. In particular it is common for multi-storey buildings to have very similar layouts on each floor. In such cases it is necessary to provide more information to the filter to enable it to complete the localisation process. One option is to use some form of absolute positioning to obtain an approximate position for the user. This approach is explored in more detail in Chapter 7.

5.3.2 Scalability

The time required by the localisation filter to perform an update is $\mathcal{O}(N\log_2 N)$ due to the re-sampling step, where N is the number of particles used to represent the state. The amount of memory required to represent the state is $\mathcal{O}(N)$. Hence it is clear that there exists a limit on the number of particles if the filter is to execute in real-time on a given piece of hardware. It is therefore important to establish how many particles are required to reliably determine the user's position. If many particles are used then they will be distributed with a high density over the floor polygons that define the environment. Conversely if few particles are used then the particle density will be low. Using fewer particles will allow the filter to execute on lower grade hardware, however the probability that there will be a particle whose state is within some distance of the user's true position and orientation is also lower. One of two things may occur if there are no particles sufficiently close the user's true state at some point during localisation:

- The localisation process can fail before the particles converge to form a single cluster. This happens if all of the particles are found to violate wall constraints in a given step update.
- The particles may converge to a location that does not correspond to the user's true position. This case arises when the user has not passed a sufficient number of asymmetries for his position to be uniquely determined, however all but one of the possible positions (including the one corresponding to the user's true location) are not represented due to an insufficient density of particles. Note that all particles at the incorrect localisation will violate environmental constraints if the user later passes an asymmetry that would otherwise have eliminated that position.

In order to determine the number of particles required for reliable localisation in the William Gates building, the localisation filter was tested on 10 sets of data recorded as a user walked along 10 different paths through the building. Each path was known to pass a sufficient number of asymmetries such that the user's position could be uniquely determined[3]. Hence the result of each execution was either a successful localisation or a failure (i.e. all particles were found to violate constraints). For each success the resulting particle cloud was visually inspected to ensure that it had converged to the user's true final position. This always occurred in practice because convergence to an incorrect location was always followed by subsequent failure before the end of the run.

The filter was executed 10 times on each log for each of a number of different particle counts ranging from $25,000$ to $1,400,000$. Figure 5.7 shows the overall failure rate as a function of

[3]This was determined by processing each log using 2 million particles (which is well in excess of the number required) and checking that the particles converged to form a single cluster around the user's true position.

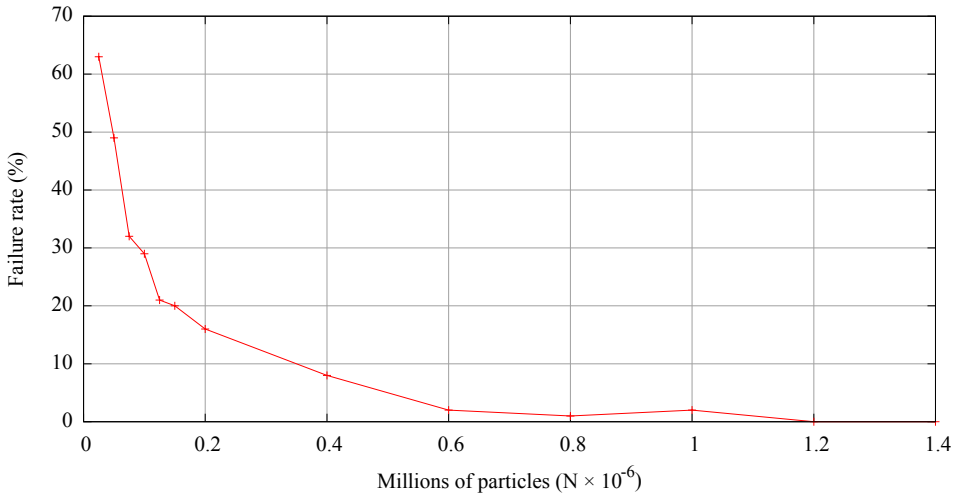


Figure 5.7: The percentage of localisation failures as a function of the number of particles used.

the number of particles used. At least $1,200,000$ particles were required for reliable localisation in all cases. Figure 5.8 shows the mean time that was required to process each step event as a function of the number of particles used. These results were obtained using a Java implementation executing on a $2.6\,\mathrm{GHz}$ desktop PC. Note that the time required for each update scales worse than the theoretical complexity of $\mathcal{O}(N\log_2(N))$ for large N. The variance in the time required to perform an update also increases. It is believed that increased memory pressure and an increase in the frequency at which the Java garbage collector was required to execute could account for both trends.

To update the $1,200,000$ particles required for reliable localisation in the William Gates building took the tested implementation on average $2.34\,\mathrm{s}$, however a user moving at a brisk walk can generate around 55 step events per minute[4]. Hence it is necessary to process each step event in $1.09\,\mathrm{s}$ if such a user is to be tracked in real time, which requires roughly a factor of two performance increase. This factor could almost certainly be attained by a more efficient implementation (in particular one that exploits multiple processors, to which the particle propagation and correction steps are well suited) or by running the existing implementation on more powerful hardware. Note however that the number of particles required scales linearly with the floor area of the building such that the particle density remains constant. Hence even with a faster implementation or more powerful hardware there will still exist a limit on the size of a building after which real-time tracking is not feasible. It is therefore desirable to find ways to reduce the computational cost of localisation without reducing the filter's reliability. One way of doing this is to dynamically vary the number of particles as the uncertainty in the user's position changes, as will be discussed in Section 6.1. This approach means that the

[4] 55 step events correspond to 110 steps taken by the user, since events are generated for only one of the user's feet.

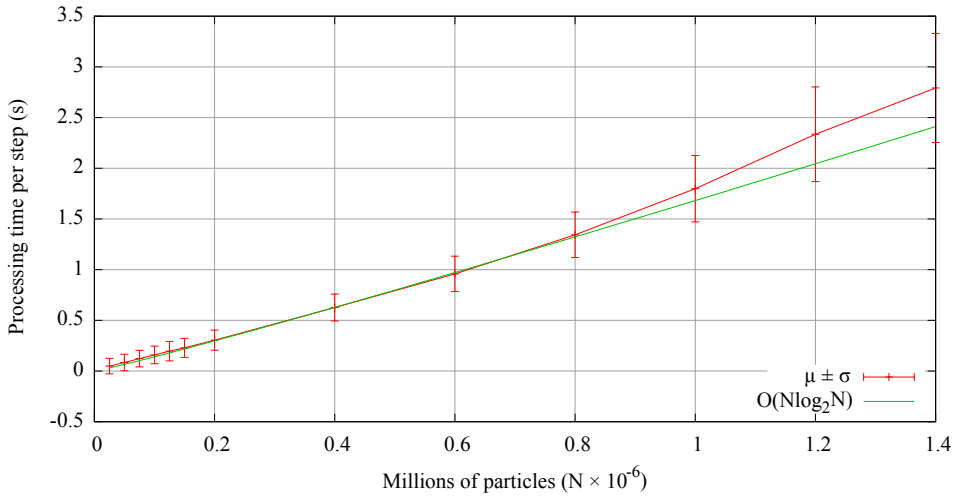


Figure 5.8: The average time required to process a step event when different numbers of particles are used. These results are for a Java implementation running on a 2.6 GHz desktop PC. The green line indicates the theoretical complexity $\mathcal{O}(N\log_2 N)$ (fit to the first nine data-points).

cost of an update is reduced with each step event. Another method of reducing the computational cost is to use another positioning system to provide a rough estimate of the user's initial position. The prior can then consist of particles generated only in the area surrounding the estimate rather than throughout the building. This approach is discussed in Chapter 7.

5.4 Tracking

Once the cloud of particles maintained by the localisation filter has converged to form a single cluster (hopefully corresponding to the user's true position) the problem solved by the filter becomes one of tracking rather than of localisation. Each time a step event is received during the tracking phase the effect of the particle propagation step is to move the particle cloud to the user's new position, expanding it slightly to model the uncertainty in the event. This expansion is limited by the correction step, since particles that have violated environmental constraints are assigned zero weights and are therefore removed during re-sampling. By continually enforcing such constraints the drift that would otherwise accumulate in the position estimates is limited, allowing the localisation filter to track a user indefinitely as they walk through the environment.

There are several ways to provide position estimates to location-aware applications during tracking. The set of particles generated by the localisation filter at a given time t represents a probability distribution for the user's state at that time. This set could be provided directly, allowing applications to make full use of the distribution that it represents (or even display

the particle cloud directly to the end user). Although this approach maximises the amount of information provided to the application, most location-aware applications only require a simple description of the user's location. For example applications could be provided with an interface to query the user's *most likely* state and optionally a simple description of the error, such as a single variance or an error ellipse.

This section describes a simple algorithm for computing an estimate of the user's position (X_t, Y_t, Z_t) and orientation Θ_t during tracking. The accuracy of the estimates is then tested by comparison with the Active Bat absolute positioning system. Note that the problem of detecting when the cloud of particles has converged to allow the computation of such estimates is not considered here, but will be discussed in Section 6.2.

The user's position (X_t, Y_t, Z_t) at time t is estimated as the weighted average of the particles, given by

$$X_t = \frac{\sum_{i=1}^{N} w_t^i \cdot x_t^i}{\sum_{i=1}^{N} w_t^i} \tag{5.32}$$

$$Y_t = \frac{\sum_{i=1}^{N} w_t^i \cdot y_t^i}{\sum_{i=1}^{N} w_t^i} \tag{5.33}$$

$$Z_t = \frac{\sum_{i=1}^{N} w_t^i \cdot z_t^i}{\sum_{i=1}^{N} w_t^i} \tag{5.34}$$

where N is the number of particles in the cluster. Slightly more care must be taken when calculating an average heading Θ_t. The naïve approach of adding up the weighted angles and then dividing by N will not always give a sensible answer. For example consider two equally weighted particles with headings

$$\theta_t^1 = -\pi + \epsilon \tag{5.35}$$

$$\theta_t^2 = \pi - \epsilon \tag{5.36}$$

where ϵ is small. Intuitively the average heading should be $-\pi$ (or equivalently π), however the value calculated by the naïve approach is 0 (i.e. the opposite direction). A better approach is to calculate the average heading vector (h_{tx}, h_{ty}) and then use it to determine a value for Θ_t [84]:

$$h_{tx} = \sum_{i=1}^{N} w_t^i \cdot \cos\theta_t^i \tag{5.37}$$

$$h_{ty} = \sum_{i=1}^{N} w_t^i \cdot \sin\theta_t^i \tag{5.38}$$

$$\Theta_t = \operatorname{atan2}(h_{ty}, h_{tx})\,. \tag{5.39}$$

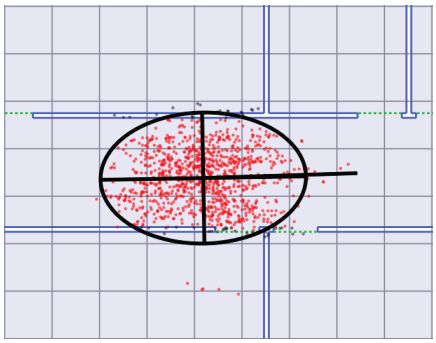


Figure 5.9: The 95% error ellipse calculated for a cloud of particles. The long black line indicates the heading Θ.

The most likely state of the user is then provided to applications as the state $(X_t, Y_t, Z_t, \Theta_t)$. If required, a simple error estimate such as an error ellipse can also be calculated, as shown in Figure 5.9. Details regarding the calculation of error ellipses can be found in Appendix E. Note that the use of a weighted average differs from the usual approach when using particle filters, in which the state of the particle with the largest weight is taken to be the most likely state. For the localisation filter this approach is not sensible since all of the particles in the cluster are likely to have very similar (if not identical) weights. Hence the state of the particle with the largest weight will effectively be a random position within the cluster.

Figure 5.10 returns to the running example used throughout this thesis to illustrate the ability of the localisation filter to track the absolute position of a pedestrian, continually correcting for errors in the individual step events generated by the PDR filter. Note that the localisation filter is not able to track the pedestrian from the very beginning of the example, since it must first localise the user as described in Section 5.3. Note also how the path of positions calculated by the localisation filter is constrained to the map of the environment throughout tracking. The final position calculated by the localisation filter has an error in horizontal position of $0.32\,\mathrm{m}$, compared to horizontal drift of $9.31\,\mathrm{m}$ incurred by the PDR filter alone. The PDR filter also incurred a vertical displacement error of $-0.48\,\mathrm{m}$, which the localisation filter corrected by enforcing the constraint that all particles are constrained to lie on floor polygons.

5.4.1 Tracking accuracy

To evaluate the accuracy of the localisation filter during tracking it was compared to positioning data obtained from the Active Bat system, which is installed in one wing on the second floor of the William Gates building. To do this an Active Bat was attached to the foot-mounted IMU worn by the user. The user then walked throughout the building, making multiple separate passes through the wing of the building in which the Active Bat system is deployed. Positions from the Active Bat system were logged by a desktop machine that was connected to the system over a wired LAN. Raw data from the IMU were logged by a UMPC carried by

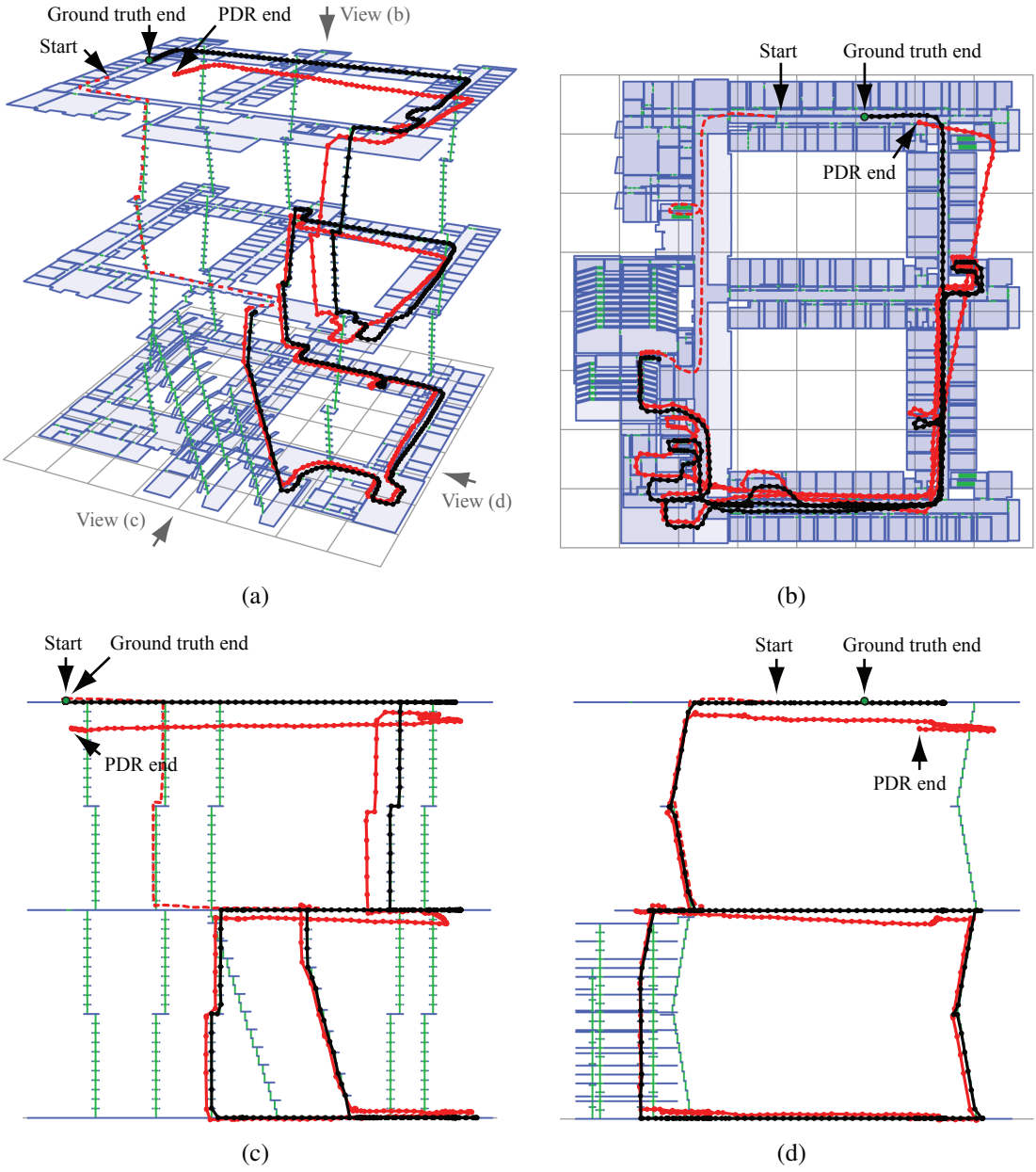


Figure 5.10: A path of step events calculated by the PDR filter as a user walked throughout the William Gates building (red) and the corrected path calculated by the localisation filter (black). The dashed portion of the red line corresponds to step events during localisation (i.e. before the filter had determined the user's position). Figure (a) is annotated with the view points of Figures (b-d). Grid size $10\,\mathrm{m}^2$. Diagrams are exploded 10x in the z-axis.

the user and post-processed on a desktop PC to obtain a set of positions estimated by the localisation filter. For each run of the localisation filter the particle cloud was initialised around the user's known initial position (i.e. it was assumed that the user had already been successfully localised).

To compare the measurements obtained from each system it was necessary to ensure that the two logging machines where time-synchronized. This was achieved using Chrony[5], which synchronises machines using the Network Time Protocol (NTP) and additionally constructs locally stored models for each clock, which are used to minimise clock drift when the machines cannot contact one another. The two machines were synchronized over a wired LAN before each data collection run. Chrony estimated a difference of at most $2\,\mathrm{ms}$ between the two clocks.

During each run various factors can cause synchronisation issues to arise. Firstly, the two clocks can be expected to drift apart slightly despite the corrections applied by Chrony. To minimise this effect each data collection run was limited to at most 15 minutes, during which the drift between the two clocks would be of the order of milliseconds. Secondly, small delays between each measurement being made and subsequently being timestamped by the logging application can have an effect. Active Bat data must also propagate over a wired LAN before reaching the logging machine. To mitigate these issues positions from the Active Bat system were only matched to localisation filter positions if they were timestamped within a stance phase, as detected by the PDR filter. This approach allowed for drift of up to $10\,\mathrm{ms}$ between the two data-sets without affecting the matched positions (since the user's foot is stationary during the stance phase and during a $10\,\mathrm{ms}$ window either side). This is an order of magnitude greater than the maximum time-stamping error that could be expected.

5.4.1.1 Filtering Active Bat positions

During testing it was found that the Active Bat system occasionally reported spurious positions, either as a result of ultrasonic noise in the environment or reflected signals from the bat attached to the user. In order to reduce the probability of such positions being included in the comparison, positions obtained from the Active Bat system were filtered before being considered for matching. The filter performed a number of checks to attempt to detect and discard spurious measurements. A position was discarded if:

- It was not located within the building.
- It was below the floor level of the wing within which the Active Bat system is deployed.
- It was greater than $10\,\mathrm{cm}$ above the level of the floor. This discards measurements

[5] http://chrony.tuxfamily.org

that are not consistent with the user's foot being grounded. These may be spurious measurements calculated when the user's foot was actually grounded (which should be discarded), or non-spurious measurements calculated when the user's foot was not grounded (in which case they should not be considered for matching anyway).

- It was calculated from fewer than five time-of-flight measurements. When five or more measurements are available the Active Bat system is able to perform consistency checks that attempt to identify and exclude[6] spurious time-of-flight measurements (i.e. those caused by signal reflections or ultrasonic noise) from the final position calculation. Hence the position calculation is more robust and therefore less likely to generate a spurious position when at least five measurements are used.

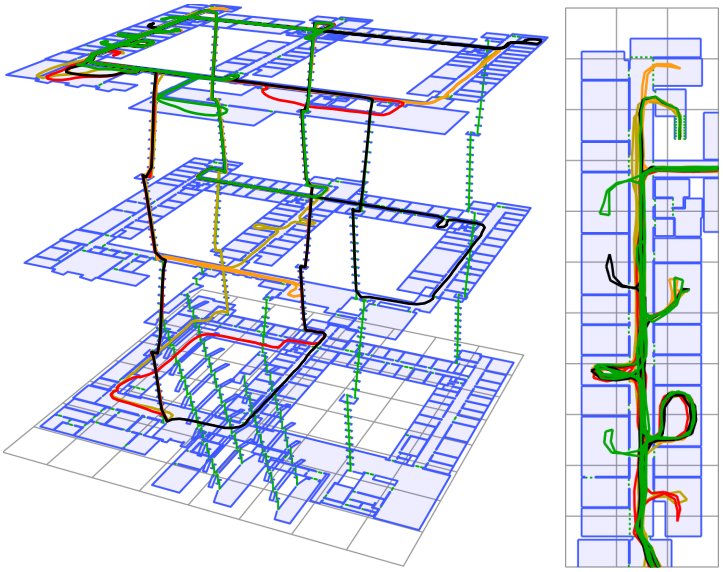
- A ray from the calculated position to the position of a receiver whose measured time-of-flight was used to calculate the position intersected a wall.

The last two conditions make use of additional information provided by the Active Bat system with each calculated position. This additional information includes the individual time-of-flight measurements, the position of each corresponding receiver and whether each time-of-flight was accepted (i.e. used to calculate the position) or rejected (i.e. classified as noise or a reflected signal) by the multi-lateration algorithm. Note that whilst some non-spurious measurements may be discarded by the filter, this will not affect the results beyond reducing the number of data-points obtained (i.e. the number of matched positions).

5.4.1.2 Results

In total data obtained during five walks were analysed. Each walk was between 10 and 15 minutes in duration and consisted of the user walking through significant portions of the building (visiting at least two of the building's three floors), as shown in Figure 5.11(a). As with localisation, using too few particles resulted in failures where all particles were found to violate environmental constraints. To determine how many particles were required the logs obtained from the five walks were processed multiple times using different numbers of particles. The number of particles used was varied from $N = 50$ to 500 in increments of 25 and then up to 1000 in increments of 100. Each walk was processed 20 times for each value of N, giving a total of 100 runs of the filter at each particle count. As expected the number of failures decreased as the number of particles was increased, as shown in Figure 5.12. No failures were observed when 275 or more particles were used. Note that this is a tiny fraction of the number required for reliable localisation and hence the $2.6\,\mathrm{Ghz}$ test machine was easily able to track a user in real time.

[6]Spurious time-of-flight measurements are usually successfully identified and excluded provided that a greater number of (and at least four) non-spurious measurements are obtained.



(a) (b)

Figure 5.11: (a) Paths extracted by the localisation filter for five different walks through the William Gates building. $N = 500$ particles were used. Grid size $10\,\mathrm{m}^2$. (b) The paths through the upper left wing of the building in which the Active Bat system is installed. Grid size $5\,\mathrm{m}^2$.

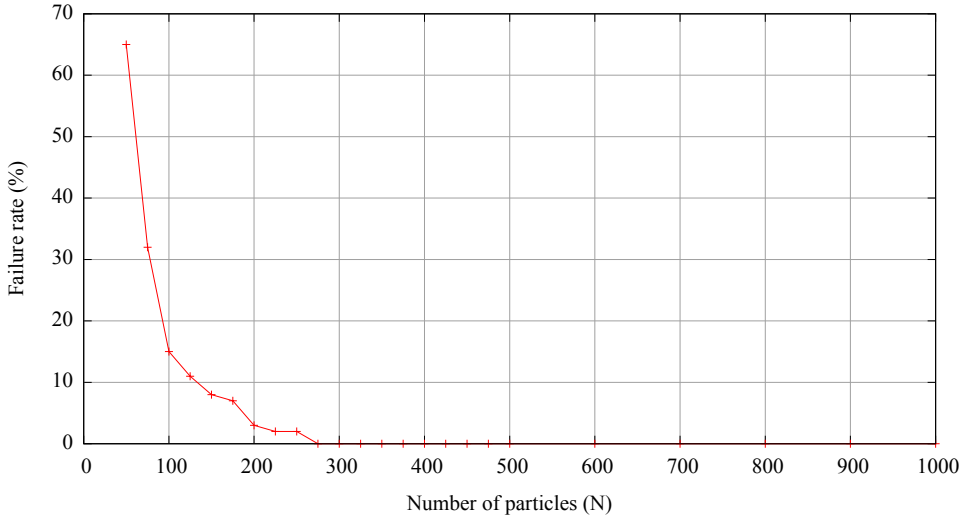
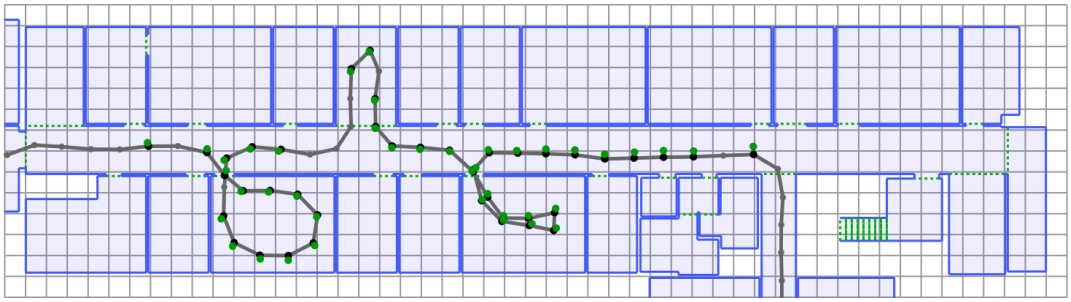
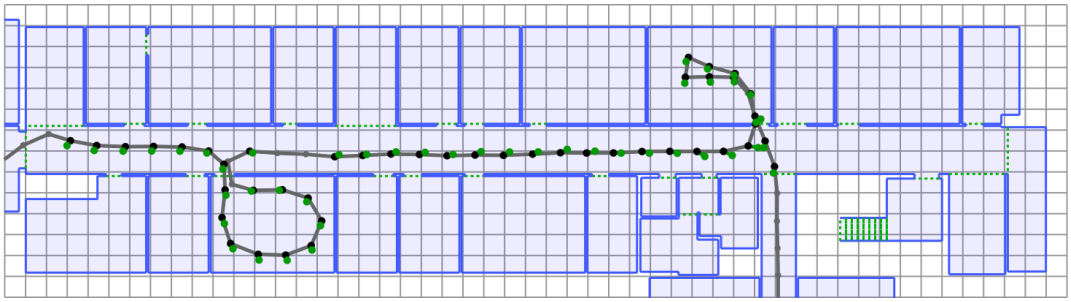


Figure 5.12: The percentage of failures during tracking as a function of the number of particles used.



(a)



(b)

Figure 5.13: Two traces through the area covered by the Active Bat system generated using $n = 500$ particles. The paths calculated by the localisation filter are shown as grey lines. Matched positions obtained from the Active Bat system are shown as green dots, each of which is connected by a line to the corresponding position obtained from the localisation filter. Grid size $1\,\mathrm{m}^2$.

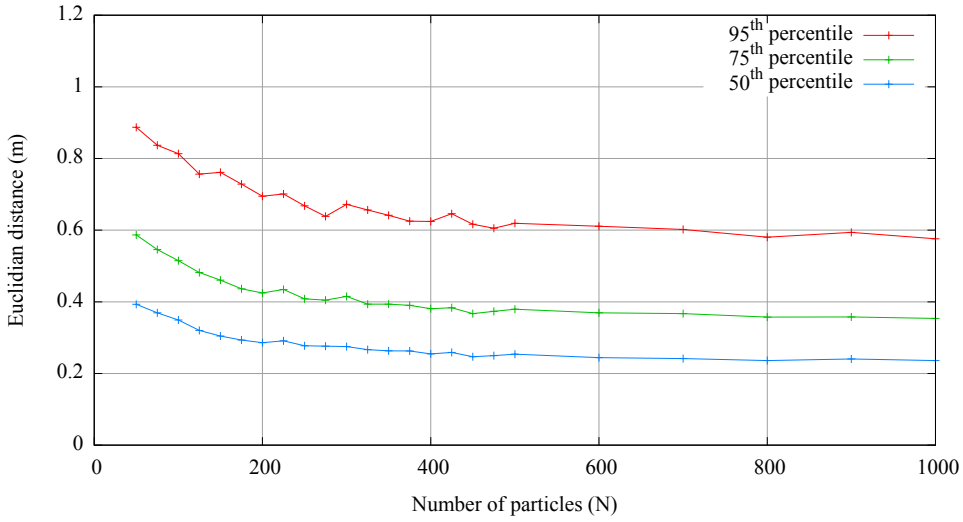


Figure 5.14: Euclidean distances between matched Active Bat and localisation filter positions. The 50^{th}, 75^{th} and 95^{th} percentiles are shown.

During each walk the user made multiple passes through the wing of the building in which the Active Bat system is deployed, as shown in Figure 5.11(b). Figure 5.13 shows two such passes together with matched positions obtained from the Active Bat system. Figure 5.14 shows the 50^{th}, 75^{th} and 95^{th} percentiles of the Euclidean distances between matched positions obtained from all *successful* runs made at each particle count. A minimum of $5,820$ distances were calculated for a given particle count ($N = 50$). $16,457$ distances were calculated for counts that observed no failures.

If the Active Bat system is taken as the ground truth (which is reasonable given the relative accuracies of the two systems) then the results shown in Figure 5.14 can be interpreted as the tracking accuracy of the localisation filter. Under this assumption the localisation filter is able to track the position of a user to within $0.38\,\mathrm{m}$ 75% of the time and $0.62\,\mathrm{m}$ 95% of the time when 500 particles are used. It is also notable that using more particles than is necessary to achieve reliable tracking did not significantly improve the accuracy of the system. One other important observation is that by enforcing environmental constraints, the localisation filter is not able to calculate paths that would have required the user to 'jump' through walls between adjacent rooms. This property does not hold for the majority of existing indoor location systems, which do not make use of such constraints. For example a WiFi location system that does not enforce environmental constraints may position a device on different sides of a wall in adjacent updates due to small fluctuations in received signal strength measurements. Note that there is no fundamental reason why such systems are unable to enforce environmental constraints (indeed there are several examples that do [85, 86]). It is simply that most existing indoor location systems have not been designed to do so.

5.5 Conclusions

In this chapter a localisation filter has been developed that combines relative movement measurements from a PDR system with knowledge of building constraints. The particle filter implementation that has been presented is able to determine the absolute position of a pedestrian in an indoor environment, provided that they pass a sufficient number of environmental symmetries. The filter is then able to track their position to an accuracy of $0.62\,\mathrm{m}$ 95% of the time in a typical office environment. This level of accuracy is suitable for many indoor location-aware applications. In particular it is sufficient to allow an application to determine which objects in the environment are close enough to the user for a physical interaction to take place.

Although the basic localisation filter has been successfully demonstrated there remain a number of outstanding problems that need to be solved in order to develop a functional tracking system. In particular localisation and tracking have until now been considered separately,

however in practice a localisation problem turns into a tracking problem dynamically as the particles converge to form a single cluster. It is necessary to detect when this occurs, since only then does it make sense to calculate state estimates using the algorithm outlined in Section 5.4. Furthermore the number of particles required for tracking is only a tiny fraction of the number required for localisation. Thus it is desirable for an implementation to dynamically reduce the number of particles as uncertainty in the user's position decreases. Algorithms for both detecting convergence and dynamically varying the number of particles will be described in Chapter 6.

There are also two issues that have been identified regarding the localisation performance of the filter:

- Symmetry of the environment can delay or prevent the convergence of the particles to a single cluster corresponding to the user's true position. Rotational and translational symmetry of this kind is often exhibited by buildings, in particular multiple floors of a building often have very similar layouts.
- A large amount of memory and computational power is required to perform reliable real-time localisation. Furthermore both requirements increase with the size of the environment due to the fact that the number of particles required is proportional to the floor area of the building. For N particles $\mathcal{O}(N)$ memory and $\mathcal{O}(N\log_2 N)$ time is required. As a result it is difficult to scale the filter to very large buildings.

Possible solutions to these problems will be discussed in Chapter 7.

Chapter 6

Efficient localisation and tracking

In the previous chapter the problems of localisation and tracking were considered separately, however in practice a localisation problem dynamically changes into a tracking problem when the particles become clustered around a single position. Considering localisation and tracking together raises two issues. Firstly the number of particles required for tracking is only a tiny fraction of those needed during localisation. It is therefore desirable to dynamically reduce the number of particles as the user's location becomes more certain. Secondly, it is necessary to detect when the particles have converged to form a single cloud of limited extent, because only then does it make sense to extract the user's position as described in Section 5.4. This chapter explores solutions to both problems. Section 6.1 describes dynamic re-sampling, a technique that automatically varies the number of particles during both localisation and tracking. Section 6.2 outlines a clustering algorithm that is able to detect when the particle cloud has converged to a single location.

The second half of this chapter focuses on the performance of the localisation filter. Section 6.3 explores how the filter performs in different environments and when provided with an inaccurate map. The filter's error characteristics are also investigated. The effect of different user motions on the accuracy of the filter is considered in Section 6.4.

6.1 Adaptive re-sampling schemes

The number of particles needed to represent $\mathrm{Bel}(\boldsymbol{x}_t)$ depends on the *complexity*[1] of the distribution, which can vary drastically over time [87]. For global localisation problems the complexity of the uniform prior (in which the tracked object is equally likely to be positioned

[1] In this context *complexity* is a vague notion describing how difficult it is to represent a given distribution. It is not a well defined mathematical quantity.

anywhere in the environment) is very high, whilst the complexity of the localised distribution after convergence is relatively low.

In the related field of robot localisation several schemes have been proposed for dynamically varying the number of particles based on the notion of distribution complexity. One possible approach is to update the particles as normal (i.e. by applying the propagation and correction steps) before using some criterion to determine how many particles should be generated during the re-sampling step. Unfortunately this approach does not perform well when the number of particles needs to be increased from one step to the next (which occurs when a measurement introduces a lot of uncertainty). Consider an example in which the previous state consists of 10 particles but where 50 are needed to adequately represent the new state. Applying the update algorithm to the 10 particles from the previous state and then re-sampling 50 times results in a set of 50 particles as required, however they can only have at most 10 distinct states because the re-sampling step simply copies states from the updated particles. Hence the particles suffer from a lack of state diversity and so do not optimally represent the underlying distribution.

Adaptive schemes developed by the robot localisation community solve the problem described above by modifying the filter so that updating the state no longer consists of an update (i.e. propagation and correction) step followed by a re-sampling step. Instead new particles of the form $\langle \boldsymbol{x}_t, w_t \rangle$ are generated one at a time by first re-sampling from the previous state

$$\hat{\boldsymbol{x}}_{t-\delta t} \sim \mathcal{S}_{t-\delta t} \tag{6.1}$$

and then applying the propagation and correction steps in the usual way

$$\langle \boldsymbol{x}_t, w_t \rangle \leftarrow \mathrm{Update}(\{\hat{\boldsymbol{x}}_{t-\delta t}, 1\}, \boldsymbol{e}_t)\,. \tag{6.2}$$

Re-sampling is implemented using a binary search into the particle weights as described in Section 5.2.3. Since the particles are re-sampled in proportion to their weights, they are assigned equal weights (nominally equal to one in Equation 6.2, although any positive constant could be used) before being updated. The process of generating particles continues until the adaptive scheme determines that a sufficient number of particles have been generated (discussed below). The particle weights are then normalised to sum to one before the set of particles is returned as the new state $\mathcal{S}_t$. This approach ensures that the number of particles updated in each step is equal to the number of particles that are actually required, and that the state of each particle is generated independently.

The job of an adaptive scheme is to determine when a sufficient number of new particles have been generated using the steps described above. Two such schemes are described in Sections 6.1.1 and 6.1.2. Note that if the cost of checking whether a sufficient number of particles has

been generated is $\mathcal{O}(1)$ then the computational complexity of generating $\mathcal{S}_t$ using an adaptive scheme is $\mathcal{O}(N_t \log_2 N_{t-\delta t})$, where N_t and $N_{t-\delta t}$ are the number of particles in $\mathcal{S}_t$ and $\mathcal{S}_{t-\delta t}$ respectively.

6.1.1 Likelihood-based adaptation

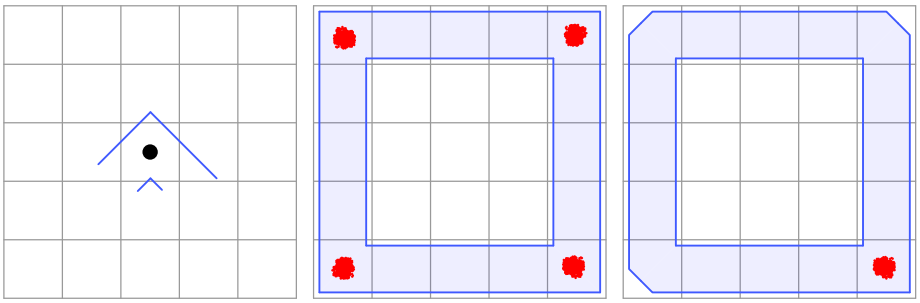
Likelihood-based adaptation generates particles until the sum of the updated particle weights exceeds a pre-specified threshold η. In other words particles are generated until

$$\sum_{i=1}^{n} w_t^i > \eta \tag{6.3}$$

is true, where n is the total number of particles generated up to that point. Hence large numbers of particles are generated in steps where the measurements do not closely match the estimated state. When the measurements are consistent with most of the particles far fewer are generated. This approach was originally applied to localise robots equipped with laser range finders [57]. In this example measurements from the laser range finders were usually less consistent with the state when there was a large amount of uncertainty in the robot's state (i.e. its position and orientation). As a result more particles were generated during the early stages of the localisation process relative to when the robot's state had been determined.

One problem with likelihood-based adaptation is that it continues to generate particles indefinitely (or in practice, until the available memory is exceeded) when a localisation failure occurs. In such cases every particle is found to violate an environmental constraint and is assigned a weight of zero, meaning that the threshold η is never reached. It is preferable for the filter to fail gracefully in such cases, for example by reporting an error that could in turn cause the localisation process to be restarted. This can be achieved in practice by introducing a maximum limit on the number of particles. If the limit is ever reached then a simple test can be used to determine whether a localisation failure has actually occurred (i.e. by checking whether the sum of the particle weights is equal to zero).

A far bigger problem with likelihood-based adaptation is that it relies on the mismatch between the prior and posterior distributions (which are approximated by the particles before and after the correction step) being closely correlated to the uncertainty in the state and hence to the number of particles required [87]. For localisation problems this is not always the case, as shown in Figure 6.1. The mismatch between the prior and posterior distributions is the same in both Figure 6.1(b) and Figure 6.1(c), however it is clear that the uncertainty is greater in the first case. For pedestrian localisation the mismatch between the prior and posterior distributions is also highly dependent on the length of the most recent step taken by the user. If the user takes a very small step then very few particles will be assigned zero weights due to



(a) (b) (c)

Figure 6.1: Particle clouds representing the possible position of an imaginary robot in two environments. (a) The obstructions sensed by the robot's laser range finder in each case. The dot indicates the robot's position. (b) The robot could be located in any of the four corners of this environment. (c) The robot can only be located in one corner of this environment. Likelihood-based adaptation will use the same number of particles in both cases.

wall constraints. Hence the two distributions will be very closely matched and few particles will be generated. This could occur early during the localisation process, when in fact many particles are required to represent the distribution to a suitable degree of accuracy.

6.1.2 Kullback-Leibler distance adaptation

Kullback-Leibler distance (KLD) adaptation generates a number of particles such that the approximation error introduced by using a particle-based representation of $\mathrm{Bel}(\boldsymbol{x}_t)$ remains below a specified threshold ϵ. This approach is better suited to localisation problems since it does not rely on a relationship existing between uncertainty and the mismatch between prior and posterior distributions [87].

To derive KLD-adaptation it is assumed that the true posterior distribution can be represented in the form of a multi-dimensional histogram. Such a histogram consists of a number of multi-dimensional bins of size Δ, each of which covers a portion of the state space within which the probability density is assumed to be constant. Given this assumption, the error between the particle-based representation and the true posterior can be measured as the Kullback-Leibler distance

$$K(p,q) = \sum_b p(b) \log \frac{p(b)}{q(b)} \tag{6.4}$$

between them, where b ranges over all bins and $q(b)$ and $p(b)$ are the probability densities for bin b according to the true posterior and the particle-based approximation respectively. It can be shown that if the number of particles is given by

$$n_{\mathrm{req}} = \frac{1}{2\epsilon} \chi^2_{k-1,1-\delta} \tag{6.5}$$

where $\chi^2_{k-1,1-\delta}$ is the upper $(1-\delta)$ quantile of the chi-square distribution with $(k-1)$ degrees of freedom, then δ is the probability that the distance between the two distributions is greater than ϵ [58]. Hence by choosing a small δ (e.g. $\delta = 0.01$) and generating $n \geq n_{\text{req}}$ particles it is possible to say with near certainty (99% certainty if $\delta = 0.01$) that a sufficient number of particles have been generated such that the distance between the particle-based representation and the true posterior is less than the threshold ϵ. In practice n_{req} can be computed using the approximation

$$n_{\text{req}} = \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \approx \frac{k-1}{2\epsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right)^3 \tag{6.6}$$

where k is the number of bins that have support (i.e. non-zero probability) in the true posterior and $z_{1-\delta}$ is the upper $(1-\delta)$ quantile of the standard normal distribution.

The complete KLD re-sampling algorithm is shown in Algorithm 3. The algorithm terminates when the total number of particles n is greater than or equal to both n_{req} and a minimum count $n_{\min}$. Since the true posterior distribution is unknown, k is estimated by counting the number of bins which have support *during* the re-sampling process. This estimate is updated each time a newly generated particle falls into an empty bin (line 11 of Algorithm 3). In the early stages of the algorithm the value of k (and as a result the value of n_{req}) increases with nearly every iteration, since only a few of the bins will already be occupied. As more bins become occupied the rate at which k increases slows down, at some point allowing the number of particles n to catch up with the required number n_{req} at which point the algorithm terminates. Termination is guaranteed since there are a finite number of bins and hence there exists a maximum value which n_{req} cannot exceed. Note that termination will occur after relatively few iterations when the particle cloud occupies only a small portion of the state space, since in this case the number of bins into which the newly generated particles fall is small. This means that when applied to localisation problems fewer particles are generated when the user's position has been determined (i.e. during tracking) relative to cases in which the user's position is unknown (i.e. during the early stages of localisation). Note that this matches the desired behaviour identified in Chapter 5.

KLD adaptation can also be used to generate prior distributions that contain a sufficient number of particles. The algorithm for generating a prior $\mathcal{S}_0$ is identical to that given in Algorithm 3, except that the state of each particle is sampled directly from the prior distribution $\text{Bel}(\boldsymbol{x}_0)$ (which must be provided as an input argument) rather than being calculated by applying the Update algorithm to a previous state. The weight of each particle is initially set to equal one, however these weights are normalised (lines 18 - 20 of Algorithm 3) before the prior is returned.

Algorithm 3 KldUpdState - Calculates the new state $\mathcal{S}_t$ from the previous state $\mathcal{S}_{t-\delta t}$ and the current step event $\boldsymbol{e}_t$. The number of particles in the new state is determined using KLD-adaptation.

procedure KLDUPDSTATE($\mathcal{S}_{t-\delta t} = \{\langle \boldsymbol{x}^i_{t-\delta t}, w^i_{t-\delta t}\rangle \, i = 1, ..., n\}, \boldsymbol{e}_t = (l, \delta z, \delta\theta, \phi)$)
// Initialise the algorithm
$\mathcal{S}_t \leftarrow \{\}$, $n \leftarrow 0$, $\alpha \leftarrow 0$
$n_{\text{req}} \leftarrow 0$, $k \leftarrow 0$
// Generate particles until the KLD condition is met
repeat
$n \leftarrow n + 1$
// Sample the previous state
$\hat{\boldsymbol{x}}^n_{t-\delta t} \sim \mathcal{S}_{t-\delta t}$
// Apply the propagation and correction steps
$\langle \boldsymbol{x}^n_t, w^n_t\rangle \leftarrow \text{Update}(\{\hat{\boldsymbol{x}}^n_{t-\delta t}, 1\}, \boldsymbol{e}_t)$
$\alpha \leftarrow \alpha + w^n_t$
// Insert the new particle
$\mathcal{S}_t \leftarrow \mathcal{S}_t \cup \{\langle \boldsymbol{x}^n_t, w^n_t\rangle\}$
if $\boldsymbol{x}^n_t$ is in empty bin b **then**
$k \leftarrow k + 1$
$b \leftarrow$non-empty
if $n \geq n_{\min} \cap k \geq 2$ **then**
// Update the number of desired particles
$n_{\text{req}} \leftarrow \frac{k-1}{2\epsilon}\left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta}\right)^3$
end if
end if
until $n \geq n_{\text{req}} \cap n \geq n_{\min}$
// Normalise the weights to sum to one
for $i = 1$ to n **do**
$w^i_t = w^i_t / \alpha$
end for
// Return the new state
return $\mathcal{S}_t$
end procedure

6.1.3 Application to pedestrian localisation

When applying KLD adaptation it is necessary to specify values for the bin size Δ, the bounding parameters δ and ϵ and the minimum number of particles $n_{\min}$. All four parameters affect the number of particles generated during each update and so must be assigned suitable values. In Chapter 5 it was shown that approximately $1,200,000$ particles are required for reliable localisation and that approximately 300 particles are needed during tracking in the William Gates building. Hence the parameters should be set such that the number of particles generated when applying KLD adaptation in the same environment is initially close to $1,200,000$ and is reduced to around 300 by the time the particle cloud has converged. There are many different sets of values that achieve this variation. Although it would be possible to perform an exhaustive search of the parameter space in order to find an optimal set of values (i.e. those that minimise the number of particles generated at every step whilst still performing reliable localisation and tracking), this task is not considered here. Instead suitable values for $n_{\min}$, Δ and δ are estimated using prior knowledge, allowing a suitable value for ϵ to be determined empirically.

- A suitable value for $n_{\min}$ is 300, since this was the minimum number of particles required for reliable tracking, as established in Section 5.4. Although it is almost certainly the case that fewer particles can be used at certain points during tracking, doing so is not important since the computational cost of tracking is already relatively cheap.

- The bin size Δ determines the granularity by which the particle count can be varied. If a very large bin size is chosen then n_{req} can only take on a small number of possible values when calculated using Equation 6.6. Hence the number of particles generated will jump between these values rather than being varied smoothly. If a small bin size is used then the number of particles will be continually adjusted but more memory will be required to keep track of the occupied bins. Since the particle cloud converges to have a radius of the order of metres, a bin size of

$$\Delta_x = \Delta_y = \Delta_z = 2\,\text{m} \tag{6.7}$$

  is sufficiently fine-grained for pedestrian localisation.

- It is also necessary to specify a bin heading granularity Δ_θ. In practice this value can be used to control the ratio between the number of particles for the prior distribution and the number used during tracking, with a smaller value of Δ_θ resulting in a larger ratio. This is because a smaller value of Δ_θ will result in more bins being occupied by particles in the prior, whilst roughly the same number of bins will be occupied during tracking provided that the Δ_θ is still greater than the accuracy to which the user's heading can be determined. A value of $\Delta_\theta = \frac{\pi}{6}$ was found to achieve a ratio close to the desired ratio

of $1,200,000 : 300$.

- From the derivation of KLD-adaptation it is known that δ is intended to be small. A value of $\delta = 0.01$ was chosen as it has been shown to give good results when applied to robot localisation [87, 58][2]. Having fixed all other parameters, a suitable value for ϵ can be determined empirically such that the absolute number of particles generated at each step is of the order required. A value of $\epsilon = 0.015$ was found to achieve this.

Although the values above have been derived based on observations made whilst using the localisation system within the William Gates building, it is likely that the same parameters will also be suitable for other buildings of similar architectural style. It is important to note that the number of particles in a prior generated by KLD-adaptation is dependent on the size of the building (more particles are generated for larger buildings). Hence it should not be necessary to adjust the parameters when applying the filter within buildings of different sizes. It may however be necessary to adjust $n_{\min}$ and the bin size if a different PDR system is used to provide step events to the filter. For example if the PDR system is less accurate then the particle cloud will be larger on average and as a result a larger bin size and value of $n_{\min}$ may be more suitable.

6.1.3.1 Localisation performance

Figure 6.2 shows the variation in particle count during an example localisation using the same data as shown in Figure 5.5 (page 100). The parameters were set as described above:

$$n_{\min} = 300 \tag{6.8}$$

$$\Delta_x = \Delta_y = \Delta_z = 2\,\mathrm{m} \tag{6.9}$$

$$\Delta_\theta = \frac{\pi}{6} \tag{6.10}$$

$$\delta = 0.01 \tag{6.11}$$

$$\epsilon = 0.015\,. \tag{6.12}$$

KLD-adaptation was used to select a suitable number of particles to represent the prior distribution and to adapt the number of particles during each update. The prior distribution itself was a uniform distribution over the entire floor surface. The user's initial heading was also distributed uniformly in all directions. As expected the number of particles decreased as uncertainty in the user's position was reduced. Note how the number of particles fell rapidly between points (d) and (e) as the user was quickly constrained to regions of the map cor-

[2] [87] contains a typing error in which it states that a value of $\delta = 0.99$ was used. This should have read $(1 - \delta) = 0.99$, or equivalently $\delta = 0.01$. This error was corrected in a more recent paper by the same author [58].

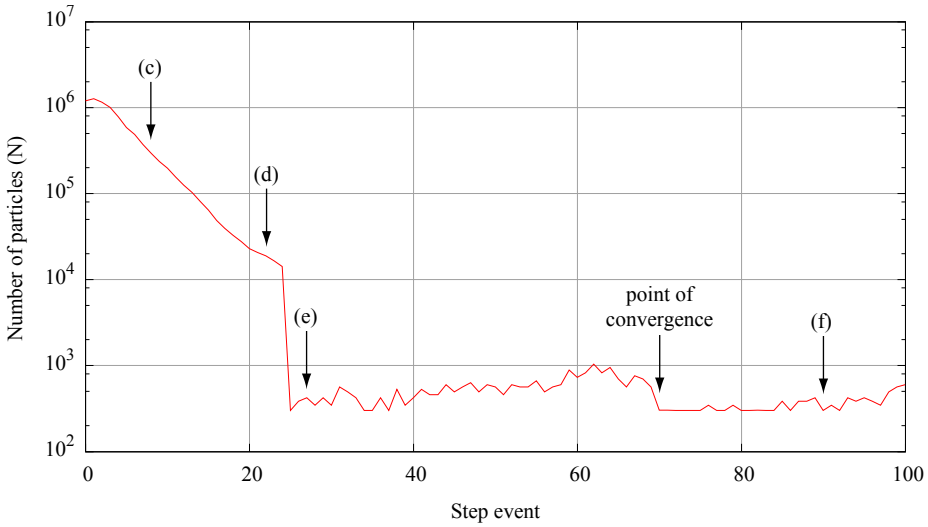


Figure 6.2: The number of particles generated by KLD-adapation for each update during an example localisation. Annotations (c-f) indicate points during the localisation which correspond to Figures 5.5 (c-f) (page 100).

responding to staircases in the environment. The particle cloud at point (f) contained 345 particles. Figure 6.3 shows the time taken to process each step event during the same localisation $2.6\,\mathrm{GHz}$ desktop PC. Although the first few steps could not be processed in real-time, the filter was able to catch up as the number of particles was reduced. In this example only the first five steps required more than $1.09\,\mathrm{s}$ to process and the filter had caught up with real-time by the 12^{th} step[3].

The reliability of the adaptive filter during localisation was tested by applying it to the same data-set used to evaluate the basic filter design in Section 5.3.2. Recall that the data-set consists of logs corresponding to 10 walks through the William Gates building, each of which is known to pass a sufficient number of asymmetries such that the user's position can be uniquely determined. Each log was processed 10 times by the filter, which successfully located the user in all 100 cases (success is defined as convergence of the particle cloud to form a single cluster around the user's known final position). Hence the reliability of the filter during localisation was not affected by dynamically adapting the number of particles.

6.1.3.2 Tracking performance

The tracking performance of the adaptive filter was measured by applying it to the same log data used to measure performance when using a fixed number of particles in Section 5.4.1.2. This dataset consisted of five logs corresponding to the user walking along five different paths in the William Gates building. Corresponding position measurements obtained using the Ac-

[3] $1.09\,\mathrm{s}$ is the average time per event that is available if the user is walking at a (brisk) rate of 110 steps per minute.

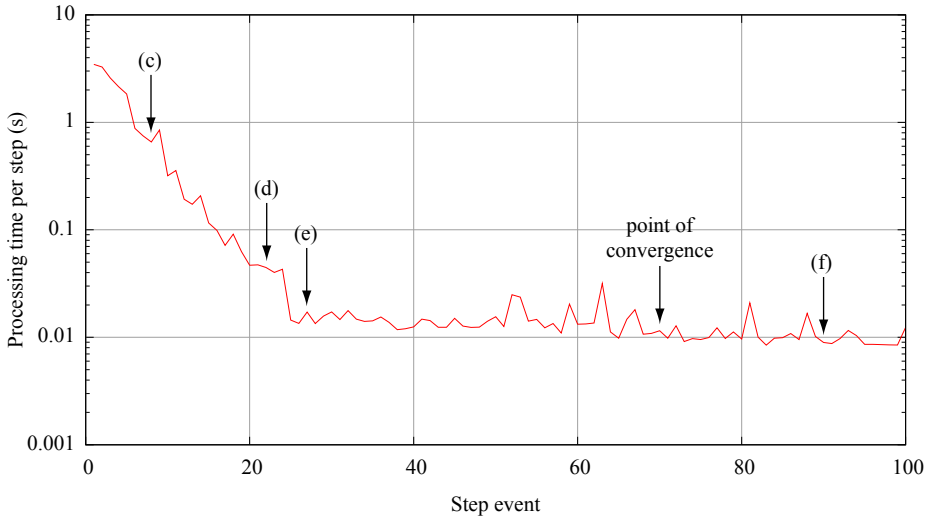


Figure 6.3: The time required to process each step event during an example localisation. These results are for a Java implementation running on a 2.6 GHz desktop PC. Annotations (c-f) indicate points during the localisation which correspond to Figures 5.5 (c-f) (page 100).

tive Bat location system were logged whenever the user was within its coverage area. The filter was used to process each of the logs 20 times, using an average of 520 particles during tracking. The minimum and maximum numbers of particles generated for a single step were 300 and 1300 respectively. No tracking failures were observed for any of the 100 runs. This was expected, since no failures were observed when processing the same log data using a fixed number of 300 particles.

Figure 6.4 shows the distribution of Euclidean distances between matched positions calculated by the adaptive filter and the Active Bat system. Recall that the Active Bat system is several orders of magnitude more accurate than the filter. Hence it is reasonable to use it as a ground truth for the purpose of assessing filter accuracy. Given this, the adaptive filter was able to track a user in the William Gates building to within $0.26\,\mathrm{m}$ 50% of the time, $0.40\,\mathrm{m}$ 75% of the time and $0.64\,\mathrm{m}$ 95% of the time. These percentiles are similar to those obtained when using a fixed number of 500 particles. Hence the use of an adaptive scheme does not noticeably improve or degrade the tracking accuracy of the system when compared to using a fixed number of particles.

6.2 Detecting convergence

During the early stages of localisation it is not possible to compute a meaningful *most likely* position for the user, since there are many particles distributed throughout the building all of which have essentially the same probability of corresponding to the user's true location. Following this early phase the particle cloud often forms a number of localised clusters, as

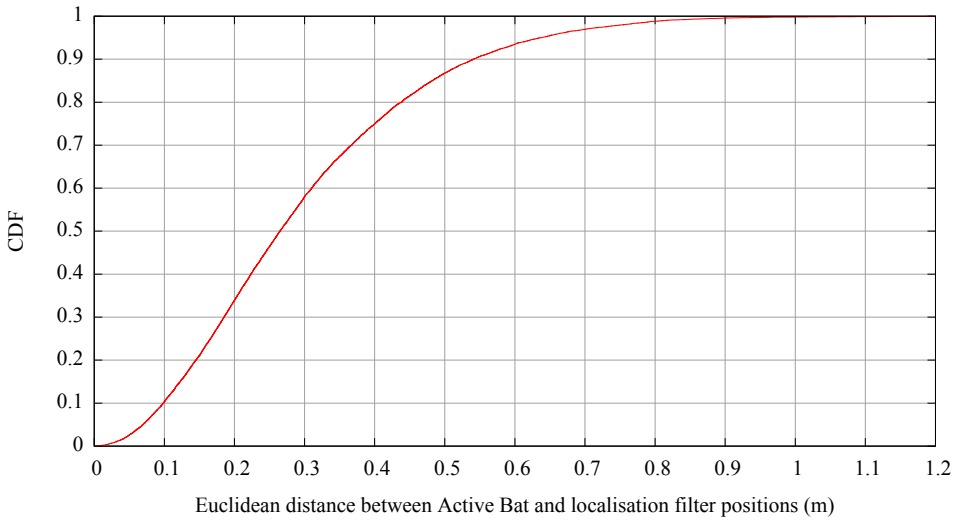


Figure 6.4: The cumulative distribution of the Euclidean distances between matched Active Bat and localisation filter positions.

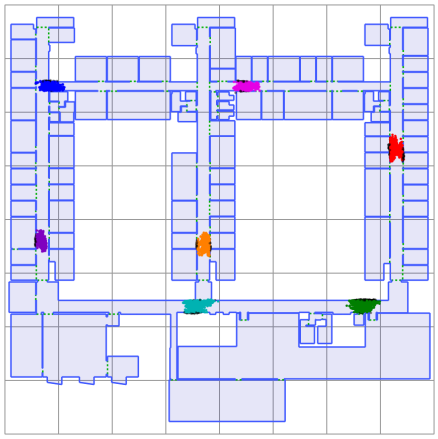


Figure 6.5: A clustered set of particles generated during localisation. The colour of each particle indicates the cluster to which it has been assigned.

shown in Figure 6.5. These clusters can be detected by applying a clustering algorithm. Over time the clusters that do not correspond to the user's true location are removed as they are found to violate environmental constraints. When all of the particles are grouped into a single cluster it is said that the particle cloud has *converged*. At this point the problem solved by the filter becomes one of tracking rather than of global localisation. During tracking the user's position can be estimated as described in Section 5.4.

The clustering algorithm used here is a variant of one originally described by Ashbrook and Starner for clustering GPS positions [88, 89]. First a particle is drawn at random and its state (x, y, z, θ) is taken as the *centre-point* of a new cluster

$$(X_c, Y_c, Z_c, \Theta_c) := (x, y, z, \theta)\,. \tag{6.13}$$

All particles that are within a specified distance (discussed below) of the centre-point are

added to the cluster. The centre-point is then updated so that it is equal to the weighted average of the cluster's constituent particles

$$X_c := \frac{\sum_{i=1}^{N} w^i \cdot x^i}{\sum_{i=1}^{N} w^i} \tag{6.14}$$

$$Y_c := \frac{\sum_{i=1}^{N} w^i \cdot y^i}{\sum_{i=1}^{N} w^i} \tag{6.15}$$

$$Z_c := \frac{\sum_{i=1}^{N} w^i \cdot z^i}{\sum_{i=1}^{N} w^i} \tag{6.16}$$

$$\Theta_c := \operatorname{atan2}(h_y, h_x) \tag{6.17}$$

where N is the total number of particles in the cluster, $(x^i, y^i, z^i, \theta^i)$ is the state of the i^{th} particle and w^i is its corresponding weight. The values of h_x and h_y are given by

$$h_x = \sum_{i=1}^{N} w^i \cdot \cos\theta_t^i \tag{6.18}$$

$$h_y = \sum_{i=1}^{N} w^i \cdot \sin\theta_t^i \,. \tag{6.19}$$

The process of adding particles that are sufficiently close to the cluster's centre-point before updating it is then repeated. This continues until there is an iteration in which no new particles are added, at which point the cluster is complete. The whole process of generating a cluster is then repeated on the remaining particles until all of the particles have been assigned to clusters.

In localisation experiments conducted in the William Gates building, it was found that adding particles to a cluster that were within an $8\,\mathrm{m}$ horizontal radius of (X_c, Y_c), a $1\,\mathrm{m}$ vertical displacement of Z_c and a $\frac{1}{4}\pi$ angle of Θ_c gave good results. Note that when all of the particles are assigned to a single cluster the centre-point of that cluster is equal to the user's estimated state when calculated as described in Section 5.4. Hence the user's estimated position and orientation are calculated *for free* by the clustering algorithm. When there are multiple clusters their centre-points can be interpreted as possible states of the user. The sum of the weights of all particles within a cluster can be interpreted as the probability of that cluster corresponding to the user's true position.

The clustering algorithm described above requires $O(N^2)$ time, where N is the total number of particles. As a result it is very expensive to cluster hundreds of thousands of particles. The solution to this problem is simply to wait until the number of particles falls below a threshold that occurs naturally during the localisation process (a suitable value for the William Gates building was found to be $8,000$). Alternatively a clustering algorithm with a lower computational complexity can be used. The important property of the algorithm described

above that makes it suitable for detecting the convergence of the particle cloud is that it does not require the number of clusters to be known in advance. Any other clustering algorithms possessing this property could equally well be used.

6.2.1 Clustered particle filtering

The clustering of particles as described above can also be used to construct a clustered particle filter, in which particles in each cluster are evolved independently according to the usual propagation, correction and re-sampling steps [90]. The idea of the clustered particle filter is to prevent the particle cloud from converging too rapidly on a single *most-likely* position when there are in fact multiple possible hypotheses consistent with the environment. This problem can arise if one hypothesis has a much larger probability than the others at a given time-step. When this occurs some of the less likely hypotheses may not be re-sampled during an update and will be lost. If the cluster corresponding to the user's true position is lost then a localisation failure will subsequently occur.

The clustered particle filter prevents premature convergence by evolving each cluster of particles independently, effectively using a separate particle filter to track each hypothesis[4] [90]. Since particles only compete with other particles in the same cluster during the re-sampling step, there is no possibility of a hypothesis being lost. The probability of each hypothesis is tracked separately at a higher level. Hypotheses are discarded when their probabilities fall to zero.

In practice failures due to the premature convergence of the particle cloud are extremely rare when applying the localisation filter in the William Gates building. This is probably due to the fact that the corridors and doorways of the William Gates building are all of similar size. Hence clusters corresponding to multiple hypotheses tend to be of similar size and have similar probabilities. As a result the probability of a given cluster dying out during the re-sampling process is very low. Note that if the environment had one very narrow corridor then many of the particles in a cluster entering that corridor would be removed, causing the corresponding hypothesis to be assigned a low probability compared to other hypotheses located in wider corridors. In this case the hypothesis in the narrow corridor could be lost prematurely. A clustered particle filter will almost certainly be more robust than the standard localisation filter when applied to environments possessing such features.

[4]Note that a single filter must be used up to the point at which the particles are initially clustered.

6.3 Dependence on environmental constraints

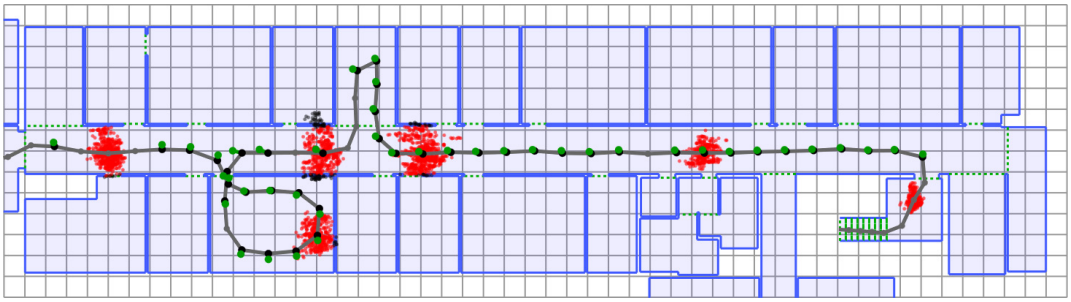
This section investigates the relationship between the 2.5-dimensional map of an environment and the performance of the localisation filter within that environment. Section 6.3.1 demonstrates how the tracking accuracy of the filter is dependent on the density of constraints. Additional constraints that can be used in environments such as an open plan office are discussed in Section 6.3.2. Section 6.3.3 describes problems that arise if the map does not accurately describe the real environment. Finally, the effect of constraints on the error characteristics of the filter are discussed in Section 6.3.4

6.3.1 Open plan environments

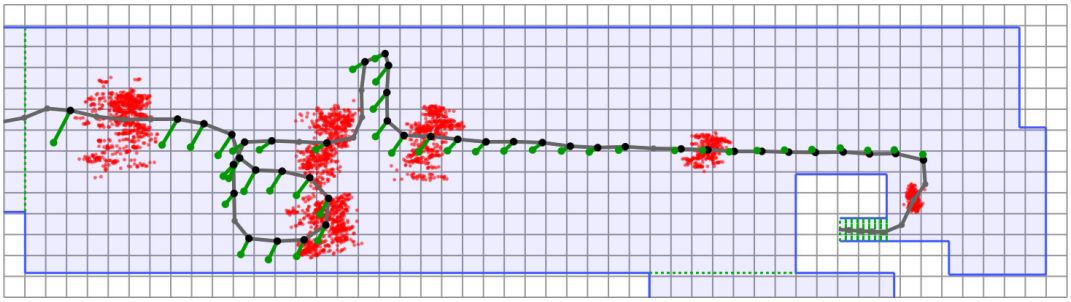
Until now the tracking accuracy of the localisation filter has been measured in a *typical* office environment; in particular the wing of the William Gates building in which the Active Bat system is installed. This is not however a fair representation of the system's accuracy in all environments. In fact the accuracy is highly dependent on the environment, since it is the environmental constraints that correct the drift errors that would otherwise accumulate in the tracked position. Hence it is expected that the localisation filter will perform better in structured environments with many constraints compared to large open plan environments where there are few.

To test the system's tracking performance in an open plan environment the filter was applied to the same log data as in Sections 5.4.1.2 and 6.1.3.2, but with a modified 2.5-dimensional map of the William Gates building. The map was modified by removing all of the interior walls in the wing of the building covered by the Active Bat system. This simulated the user following the same paths, but through a large open plan environment rather than through the office environment that actually exists. Figures 6.6(a) and 6.6(b) show the result of removing the interior wall constraints for a single walk through the modified area. In this example a heading error was propagated from the PDR filter as the user entered from the right. In Figure 6.6(a) the corridor walls constrained the particle cloud and hence prevented error from accumulating in the user's position. Without the interior wall constraints this was not possible, as shown in Figure 6.6(b). The distribution of positioning errors obtained when only exterior walls were used is shown in Figure 6.7. As expected the accuracy was degraded, with 95% of measurements accurate to within $1.58\,\mathrm{m}$ (compared to $0.64\,\mathrm{m}$ when interior wall constraints are used).

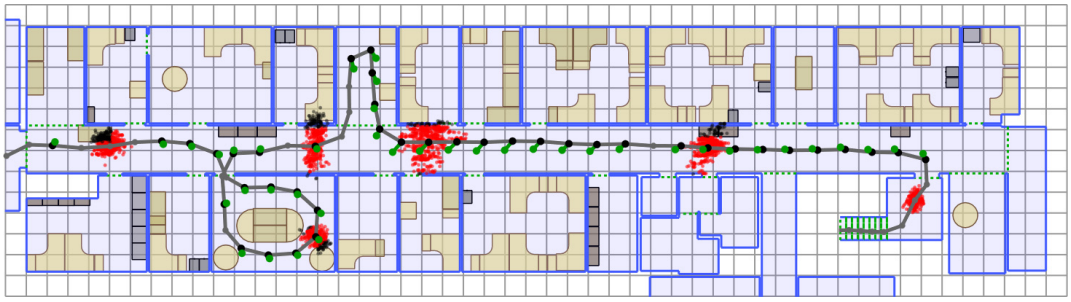
Open plan environments similar in size to the one simulated in the example above are common in many buildings. In some cases (e.g. a large hallway) the observed degradation in accuracy is unavoidable, however in others (e.g. an open plan office) there are often objects other than



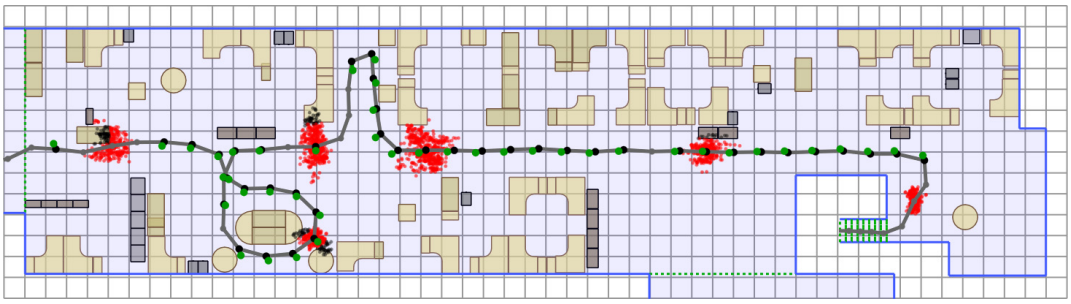
(a)



(b)



(c)



(d)

Figure 6.6: The paths calculated by the localisation filter for a single walk; (a) With interior wall constraints. (b) Without interior wall constraints. (c) With both furniture and interior wall constraints. (d) With furniture constraints but without interior wall constraints. In each case the cloud of particles is shown for every 10^{th} step. Corresponding positions calculated by the Active Bat system are shown as green dots. Grid size $1\,\text{m}^2$.

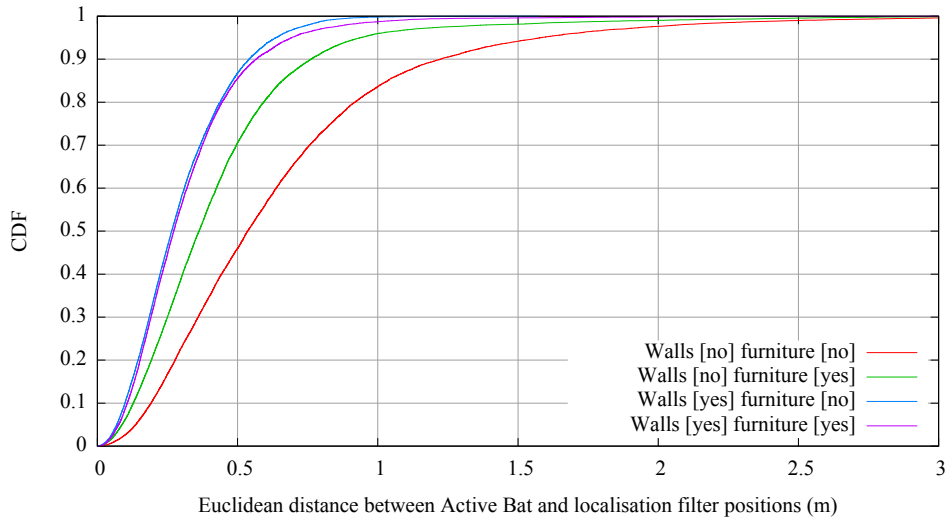


Figure 6.7: Tracking accuracy with and without interior wall and furniture constraints.

walls that can be used to constrain the user's movement. In particular items of furniture act as constraints in most open plan office environments. The use of these additional constraints is outlined below.

6.3.2 Additional constraints

Apart from walls, furniture such as desks and bookcases often limit the movement of a user through a typical office environment. If the locations of such objects are known then they can easily be used as additional constraints by the localisation filter. OpenRoomMap was developed to provide a mechanism to obtain such information [91]. OpenRoomMap is a web-based application that allows the occupants of a building to place items such as furniture onto a floor-plan, as shown in Figure 6.8. Its aim is to allow many individuals to contribute towards building and maintaining detailed models of indoor environments for use by ubiquitous computing systems. This approach of crowd-sourcing data has already proven to be successful in a number of other domains, most notably in the construction of the on-line encyclopaedia Wikipedia[5]. OpenStreetMap[6] is also using crowd-sourcing in an attempt to build an open-source map of the world [92].

It is not sensible to use all of the furniture items as constraints. Small pieces of furniture such as chairs and coffee tables are moved frequently in the physical world, whereas their positions in the OpenRoomMap model are likely to be updated far less frequently, if at all. Hence only large rarely-moved objects should be used, for example full-sized desks, cupboards and filing cabinets. The 2.5-dimensional map data-structure is easily extended to allow the representa-

[5]http://www.wikipedia.org
[6]http://www.openstreetmap.org

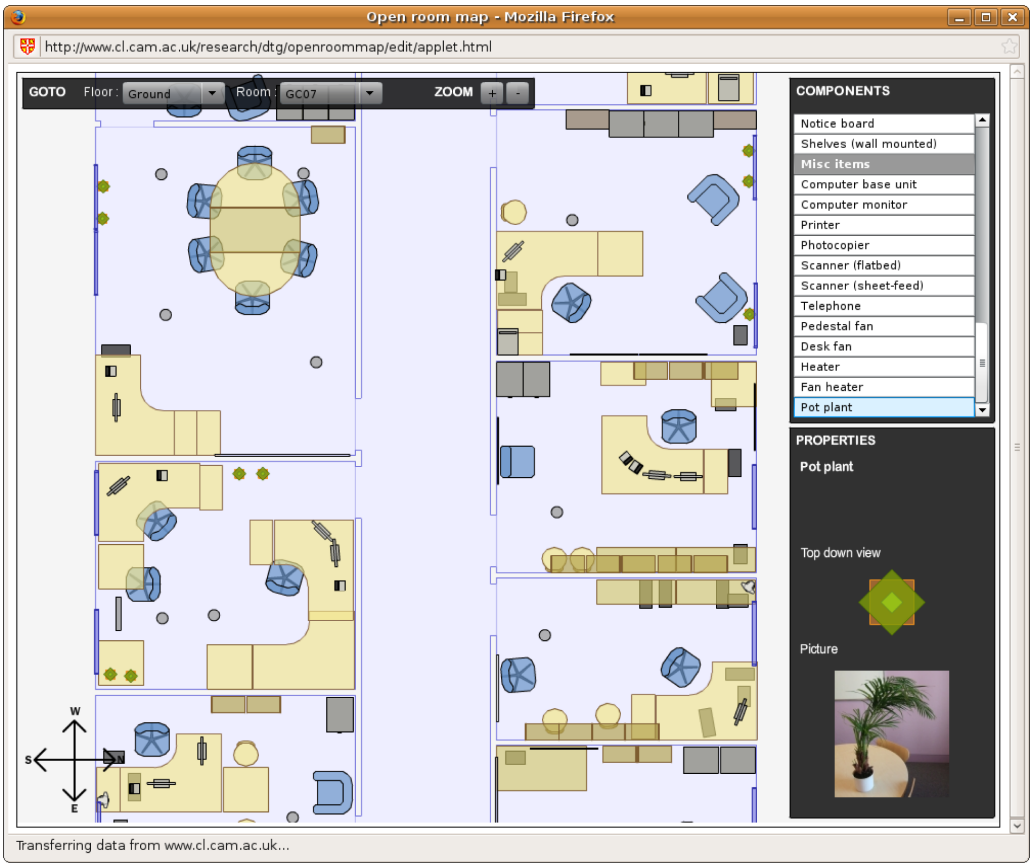


Figure 6.8: The OpenRoomMap editor.

tion of such objects by adding a list of overlapping furniture objects to each floor polygon. Note that if a furniture object spans multiple floor polygons then it should be entered into all of the corresponding lists. To enforce furniture constraints the localisation filter can simply treat the edges of all furniture objects in a floor polygon's furniture list as impassable walls belonging to that polygon.

Figures 6.6(c) and 6.6(d) show paths calculated by the localisation filter using furniture constraints with and without interior wall constraints. When used in addition to interior walls the accuracy of the calculated path was not significantly altered. In fact across all test runs the tracking accuracy of the filter was slightly worse when furniture constraints were used (although the difference is not believed to be significant), as shown in Figure 6.7. In contrast it is clear from Figures 6.6(b) and 6.6(d) that the use of furniture constraints in the absence of any interior walls significantly improved the accuracy of the calculated path. Over all test runs the use of furniture constraints reduced the 95^{th} error percentile from $1.58\,\text{m}$ to $0.94\,\text{m}$ in the case where interior walls were not used.

6.3.3 Constraint accuracy

When developing localisation filters it is often assumed that the virtual model of the environment is an accurate reflection of the real world. In practice this is only achievable in controlled research environments. The problem of inaccurate constraints is particularly likely to arise when using crowd-sourced data such as that described in Section 6.3.2. Objects may be placed at incorrect locations or may be moved (in the physical world) without being updated in the virtual model. Although these problems are particularly applicable to furniture due to its mobility, they are also true of the map itself. Building walls can also be moved, added and removed, albeit less frequently.

It is possible to describe inaccuracies in a building model in terms of missing and non-existent constraints:

1. **Missing constraint:** A constraint in the real world that is not represented in the model.
2. **Non-existent constraint:** A constraint in the model that does not exist in the real world.

Note that the third case of an incorrectly placed constraint can be thought of as a non-existent constraint in conjunction with a missing constraint at the object's true location.

The effect of a missing constraint is simply one fewer constraint available for use during localisation and tracking. During localisation this can delay or prevent localisation if removing the constraint increases the environmental symmetry of the environment, as described in Section 5.3.1. During tracking the accuracy of the filter in the vicinity of the missing constraint may be reduced. This effect is demonstrated by the experiment described in Section 6.3.1, where the interior walls of an area of the William Gates building were removed from the model in order to measure the localisation filter's tracking performance in an open plan environment.

Non-existent constraints pose a bigger problem to the localisation filter since they can cause a failure in which the entire particle cloud is found to violate the constraints. For example if a user who is being tracked walks through a wall that exists in the model but not in real life then all of the particles will be assigned zero weights. Hence the localisation filter will believe that it is impossible for the user to be positioned anywhere in the building, at which point the filter must be re-started (from scratch) or from a guess of the user's position calculated using some ad-hoc recovery algorithm. If the user violates a non-existent constraint during localisation then the problem is even worse. From that point on there will be no particles corresponding to the user's true position, however since there will still be particles located elsewhere an immediate failure will not occur. The filter will only fail when all other candidate positions have been eliminated. Note that the particle cloud may well converge before this occurs, causing the filter to report incorrect positions for the user.

In practice it is often possible to detect non-existent constraints in an environment, provided

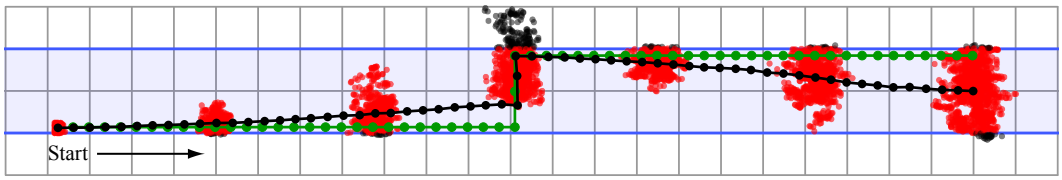


Figure 6.9: Bias errors caused by nearby environmental constraints. The path of a simulated user is shown in green. The corresponding path calculated by the localisation filter is shown in black. The particle cloud is shown for every 10^{th} step. Grid size $4\,\text{m}^2$.

that they are few in number. This can be achieved by logging which constraints are violated each time a failure occurs. If a particular set of constraints is found to account for multiple failures then it is likely that at least one of them does not actually exist. Note that this approach is not foolproof, since delayed failures as described above usually occur when the particle cloud violates a valid constraint having previously converged to an incorrect location. It is however a useful method to flag suspect constraints that can then be checked manually. In the case of crowd-sourced data the user who last edited a suspect constraint could be asked to perform the check.

Another idea that may prove useful when dealing with possible model inaccuracies is the idea of probabilistic constraints, where the probability p of a given constraint is the certainty that a corresponding object actually exists in the real world. When a particle violates such a constraint its weight is multiplied by $(1 - p)$. Hence if a constraint is certain (i.e. $p = 1$) then any particles that violate it will be assigned a weight of zero and as a result will be removed during re-sampling. Conversely if a constraint is definitely non-existent (i.e. $p = 0$) then it will have no effect. When a constraint is thought to be responsible for a failure its probability could be reduced. Hence constraints that are thought to be responsible for multiple failures will have their weights reduced over time, reducing their effect on the particle cloud during subsequent runs. The probability of furniture constraints could also decay over time to represent the fact that they may have moved, with the decay rate dependent on the mobility of the item (e.g. coffee tables are more mobile than heavy filing cabinets). The weights of such constraints could then be restored to $p = 1$ each time their positions are manually confirmed.

6.3.4 Tracking error characteristics

Although constraints are important to prevent the accumulation of drift errors over time, they also introduce bias into the positions calculated by the localisation filter. In particular if a user is located near some constraint, then the position calculated by the localisation filter will tend to be biased away from that constraint. This phenomenon is demonstrated in Figure 6.9. In this example a simulated user walks in a straight line down one side of a corridor before crossing to the other side and continuing in the same direction. The user's initial position

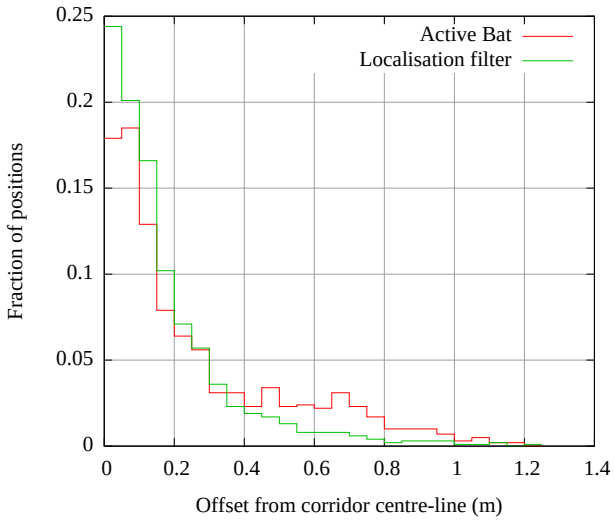


Figure 6.10: Deviations from the corridor centre-line calculated for matched positions obtained from the localisation filter and Active Bat systems.

and orientation are assumed to be known, hence the prior consists of particles only with this initial state. The user's movement was simulated by a sequence of perfect step events (shown in green) that were fed to the localisation filter. As successive step events are processed by the filter the particle cloud expands across the width of the corridor, however expansion in the other direction is prevented by the nearest wall. As a result the positions calculated by the filter become biased towards the centre of the corridor. When the user crosses to the other side of the corridor this bias is corrected, however it is re-introduced (this time in the opposite direction) as the user continues down the corridor. Note that the rate at which the bias is re-introduced is dependent on the step error model, which is itself dependent accuracy of the PDR system. If a very good PDR system is used then the particle cloud will expand slowly because each step event will carry only a small amount of uncertainty. If a poor quality PDR system is used then the particle cloud will expand rapidly and hence the bias will become noticeable after fewer steps.

A bias towards the centre-line of the corridor can also be observed in traces calculated from real measurement data, although the effect is less clear due to noise and other bias errors that perturb the raw IMU measurements. Consider the set of positions calculated by the localisation filter that were matched to corresponding Active Bat positions during the tests described in Section 6.1.3.2. Now consider the subset of these positions for which the corresponding bat positions were located in the central corridor running through the area covered by the Active Bat system. It is trivial to calculate the distance from the corridor centre-line of each position in this set. The distribution of these offsets is shown in Figure 6.10 together with the same distribution calculated from the corresponding Active Bat positions. Note that the majority of the positions obtained from both systems were close to the centre-line of the corridor. This is because a pedestrian will naturally tend to walk down the middle of a corridor unless entering

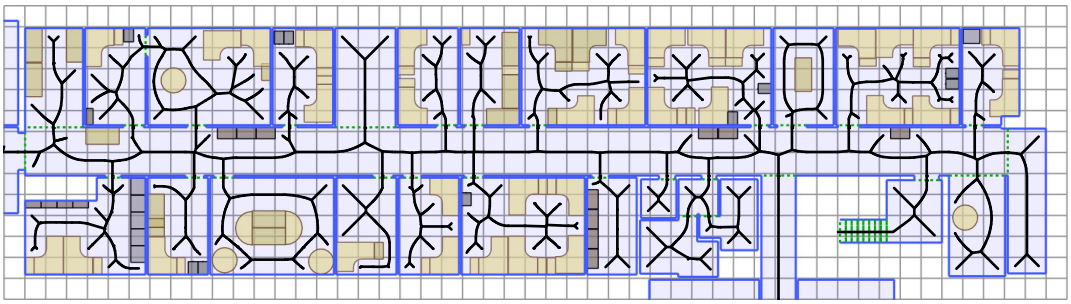


Figure 6.11: A Voronoi diagram of an office environment (generated using VRONI [93]).

or exiting a room. Comparing the two distributions, a greater proportion of the positions calculated by the localisation filter lie very close to the centre-line, whilst the Active Bat system reported more positions near to the walls of the corridor. Hence the data shows a bias in the positions calculated by the localisation filter towards the centre of the corridor.

6.3.4.1 Voronoi diagrams

Another way to think about the biasing effect of environmental constraints is to consider a Voronoi diagram of the environment. A Voronoi diagram consists of a set of points with a locally maximal distance from all surrounding objects. Such points naturally form the edges of a graph-like structure, as shown in Figure 6.11. The effect of nearby environmental constraints can be thought of in terms of introducing a bias error towards the nearest edges of the Voronoi diagram rather away from nearby constraints. Note that although it is possible to predict the bias error that will be introduced given the true position of the user, it is not possible to reverse the error since many points in free space map onto a single point on the graph. Understanding the biasing effect in this way gives weight to the idea of constraining particles to lie on the edges of a Voronoi diagram. Such an approach has previously been used to estimate the position of a user based on measurements from absolute positioning systems [85]. The main benefit is that since the particles are more constrained, far fewer are required and hence the computational resources required to run the filter are much lower. The main drawback is that it is not possible for the particle cloud to represent positions that lie near to constraints (e.g. at the edge of a corridor), however since the biasing effect makes the correct calculation of such positions unlikely in any case this limitation may not have a significant effect on the accuracy of the filter. In fact the overall accuracy may actually improve, since pedestrians have a natural tendency to move along the edges of a Voronoi diagram (i.e. staying well clear of nearby obstacles) when moving through an environment.

The use of Voronoi diagrams is not discussed further in this thesis, however it is likely that such diagrams will prove useful in future research on pedestrian localisation and tracking. In particular they are likely to be beneficial when using low quality PDR systems (for which the

biasing effect is large) or when computational resources are limited (in which case the use of fewer particles is attractive).

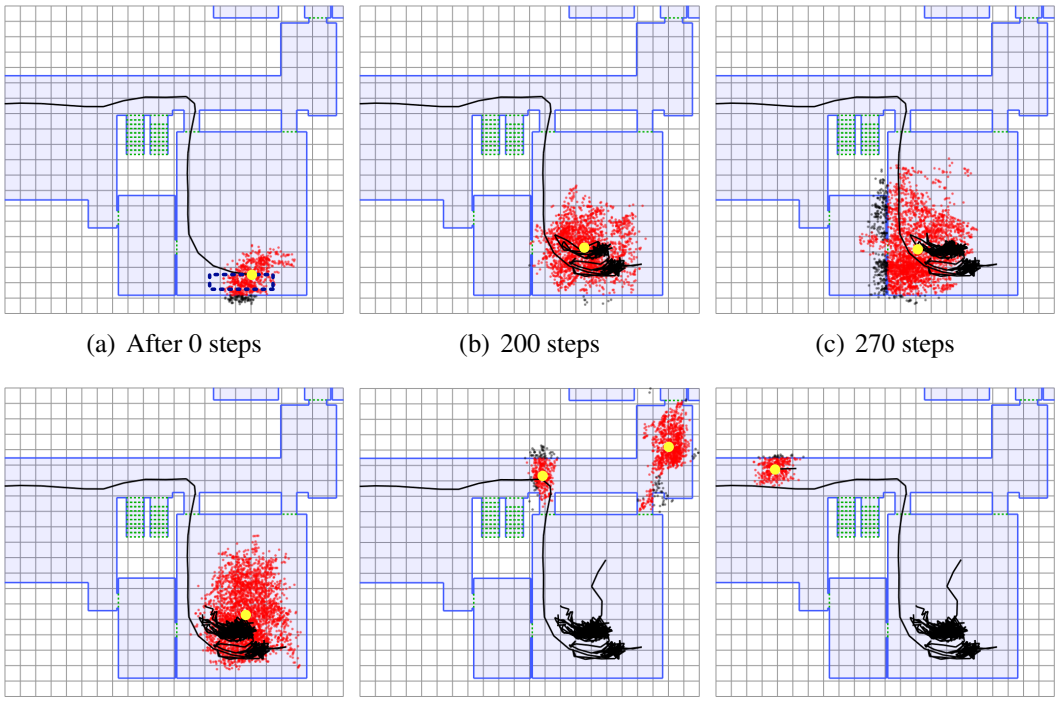
6.4 Dependence on user motion

It is not always the case that users are continually walking from one place to another, as has been the case in the examples used in Chapter 5 and Sections 6.1 and 6.3. In particular users may *loiter* in particular areas, for example when chatting in the corridor of an office environment. Figure 6.12 shows various points during an example in which the user was tracked whilst giving a 30 minute presentation. During the presentation 440 step events were generated by the PDR filter corresponding to small steps taken by the user. Most of these steps were *on-the-spot*, although a few were generated by the user moving to point out various objects on his slides, which were projected onto a nearby wall.

Figure 6.12(a) shows the user entering the room to start the presentation. The cloud after 200 step events is shown in Figure 6.12(b) . Note that the particle cloud has expanded due to the uncertainty in each of the events. A bias in the calculated position away from nearby walls is also observed, as described in Section 6.3.4. Figure 6.12(c) shows how particles propagate into a neighbouring room as the user moves to the left in order to point out an object on his slides, however by the end of the presentation the particle cloud is once again entirely within the correct room, as shown in Figure 6.12(d). As the user leaves the room two distinct clusters are formed, as shown in Figure 6.12(e). This occurs because uncertainty in the user's orientation has increased during the talk to such an extent that he could be moving towards either of the two upper exits. One of these clusters is quickly removed as the user moves further away from the room, as shown in Figure 6.12(f).

It may seem somewhat surprising that the particle cloud does not expand further than the bounds of the room (except temporarily as in Figure 6.12(c)). One might expect that the particle cloud would expand to fill a larger and larger proportion of the floor of the building throughout the talk[7]. This does not happen in practice because each particle follows a random-walk-like path due to uncertainty in each of the *on-the-spot* step events. As a result any particle that is in close proximity to a constraint is likely to violate it at some point in the future and hence be removed. To move into a different room a particle must come into close proximity with nearby constraints because the doorways are relatively narrow. Hence very few particles successfully pass into neighbouring rooms. Those that do tend to be removed over the course of multiple re-sampling steps due to the stochastic nature of re-sampling.

[7]Note that the particles would not spread to different floors due to the use of δz as a measurement in the localisation filter.



(a) After 0 steps (b) 200 steps (c) 270 steps

(d) 440 steps (e) 456 steps (f) 462 steps

Figure 6.12: Tracking a user as they give a 30 minute talk. The dashed blue box in (a) indicates the region within which the user was positioned throughout the talk. In each figure the black line shows the calculated path up to that point. Each yellow dot indicates the centre-point of a cluster of particles. Grid size $1\,\mathrm{m}^2$.

6.5 Conclusions

This chapter has described algorithms for dynamically adapting the number of particles used by the localisation filter and for detecting the convergence of the particle cloud to a single location. Both algorithms enhance the basic localisation filter and deliver important benefits when deploying such filters in practice. Adapting the number of particles using KLD-adaptation allows the filter to generate only the number of particles it requires to represent the belief at each point in time to a specified level of accuracy. The important consequence of using such an adaptive scheme is that step updates do not consume more computational resources than they require. Hence only the initial localisation phase is computationally expensive. The ability to detect when the particle cloud converges to a single location is also important, since only then does it make sense to compute the user's position as the weighted average of the positions of the individual particles.

The second half of this chapter analysed how the accuracy of the filter depends on the environment and the motion of the tracked user. The important results are summarised below.

- The accuracy of the localisation filter is dependent on the density of the environmental constraints, with a higher density of constraints leading to a better accuracy.

- Additional constraints such as those provided by furniture can be used to increase the constraint density. This approach is particularly applicable to open plan offices, which typically have few walls but many pieces of furniture.
- Inaccuracies in the map can lead to problems during both localisation and tracking. They can be characterised in terms of missing constraints and non-existent constraints. Missing constraints can delay or prevent convergence during localisation and degrade the system's tracking accuracy. Non-existent constraints pose a bigger problem since they can cause failures during both localisation and tracking.
- The positions calculated by the filter are biased away from nearby constraints, or equivalently towards nearby edges of a Voronoi diagram representation of the environment. By constraining particles to lie on such a graph it may be possible to decrease the number of particles required to track a user whilst having a minimal effect on the system's accuracy.
- Surprisingly the particle cloud does not tend to spread throughout the environment when the user takes many small steps whilst loitering in a single location. This is because particles are unlikely to pass through doorways (which are typically narrow) without subsequently violating nearby wall constraints.

Although the use of KLD-adaptation automatically reduces the computational requirements as the filter becomes more certain of the user's position, it does not reduce the initial cost (i.e. the computational cost and memory required during the first update). Hence even when varying the number of particles using such an algorithm it is not feasible to apply the localisation filter to an arbitrarily large environment. Chapter 7 will investigate methods to reduce this initial cost by using a range of additional sensors.

Chapter 7

Assisted localisation

Whilst tracking using the algorithms developed in Chapter 5 is computationally inexpensive, the cost of initial localisation is high. The adaptive schemes discussed in Chapter 6 are able to automatically reduce this cost as the user's position becomes more certain, however they are not able to reduce the cost of the first few updates during the localisation process. The computational and memory requirements for an update are $\mathcal{O}(N\log N)$ and $\mathcal{O}(N)$ respectively, where the number of particles N is initially proportional to the floor area of the building in which the system is deployed. This places a practical limitation on the size of buildings within which global localisation can be performed. Environmental symmetry can also delay or even prevent the filter from determining the user's position in a large building, as described in Section 5.3.1 and illustrated in Figure 7.1.

Both of the problems outlined above can be solved, at least to some extent, by reducing uncertainty during localisation. This is possible if some additional information is available about the user's position or orientation, which could be obtained from additional sensors or provided manually by the user. When such information is available it can be used to place additional constraints on the prior (i.e. the initial distribution of particles) and the sets of particles generated by the filter during localisation. This not only reduces the number of particles that are required, but can also reduce ambiguity caused by environmental symmetry by removing clusters of particles that are inconsistent with the constraints.

There are many ways in which the position of a user can be constrained both when generating the prior distribution and throughout the localisation process. The most suitable approach is dependent on the scenario in which the localisation system is to be deployed. In some scenarios it may be reasonable to require the user to enter their approximate location when the system is first switched on; for example when tracking military or emergency personnel as they enter a building. In such cases the users may be sufficiently motivated to provide an approximate initial position in order to enable tracking. For other usage scenarios this

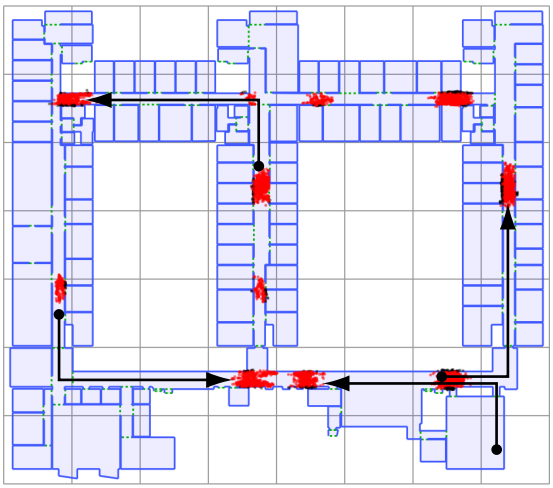


Figure 7.1: Multiple clusters formed due to environmental symmetry. Arrows indicate the paths followed by four of the clusters. The top left cluster corresponds to the user's true position. Grid size $10\,\mathrm{m}^2$.

approach is less practical. For example employees being tracked in an office environment as part of a use-of-space experiment will not benefit directly from the correct operation of the tracking system. Hence they will be less inclined to provide manual input when requested to do so. In some scenarios the user might be lost, making it impossible for them to enter their position.

An alternative to requesting user input is to make use of additional sensors in order to constrain the user's state during localisation. A wide range of sensor types could be used, some of which are listed below.

- A magnetometer can be used to estimate the user's initial heading. If the heading can be estimated to within 10° then the number of particles required to represent the prior can be reduced by a factor of 36 for a given floor area. An initial heading estimate will also resolve ambiguity due to rotational symmetry in the environment. This approach to reducing uncertainty is desirable since many commercial inertial measurement units (IMUs) already contain magnetometers. Unfortunately the use of magnetometers is not possible in environments that contain sources of strong magnetic fields. In the case of the William Gates building magnetic disturbances caused by metallic floor panels make it impossible to reliably determine the user's orientation using the magnetometers in the foot-mounted Xsens IMU, as described in Section 4.3. An alternative approach in this case would be to use a separate magnetometer mounted at a higher position on the body, such as on the hip.

- If a user is being tracked by an outdoor positioning system (e.g. GPS) prior to entering the building, then it may be possible to use the last positions calculated by such a system in order to determine the user's entry point into the building, or a set of possible entry

points if the outdoor positioning system is not sufficiently accurate. Particles in the prior distribution can then be focused around these entry points. Such an approach may be well suited to military personnel, who are often equipped with GPS receivers for outdoor use.

- In some cases it may be possible to augment the environment with a small amount of fixed infrastructure. For example some form of beacons (e.g. Bluetooth beacons or RFID tags) could be mounted near to the entry points of a building. A device attached to the user could scan for such beacons in order to determine the user's approximate initial location when entering the building. The localisation filter could then be initialised with a cloud of particles only in the vicinity of a beacon detected from the user's initial location. The number of particles required to represent the prior is independent of the size of the building when this approach is used. The main disadvantage is that a constrained prior can only be generated when a beacon is detected from the user's location. Hence if beacons are placed only at entry points to a building then they cannot be used to generate a constrained prior for a user who is already inside the building when the system is switched on.
- Wireless LANs are already deployed in many buildings. The strengths of signals received from nearby WiFi access points can be used to estimate the user's initial position and hence constrain the user's location during localisation. This approach usually requires a pre-constructed radio map of the environment or at the very least the positions of the access points in the environment. Hence such an approach may not be suitable for the emergency services scenario in which personnel are typically unable to survey the building in advance. It is however well suited for tracking in office environments. The use of WiFi access points during localisation in described below.

7.1 WiFi-assisted localisation

Where available, signals received from WiFi access points in the environment can be used to assist with the localisation process. The requirements for assisting the localisation filter differ substantially from those of traditional WiFi positioning systems. As a result there are several important differences between the algorithms described here and those used in the WiFi positioning systems described in Section 2.1.2.4. Firstly, the aim of a WiFi positioning system is to calculate the most likely position of a person or device as accurately as possible. In contrast the aim here is to assist the localisation filter by reducing the number of particles that are required and where possible resolve ambiguities that arise due to environmental symmetry. Constraining the user's position to a relatively coarse region of space is usually sufficient to achieve these goals. Furthermore it is preferable to make such regions large enough to con-

tain the user with a very high probability, rather than attempting to position the user more accurately but in a less robust manner. An important consequence of only requiring coarse location information is that it is not necessary to install a dense collection of access points in the environment. Nearly all of the WiFi positioning systems described in Section 2.1.2.4 were tested using many more access points than would have been required just to provide wireless coverage in the deployment areas. The algorithms developed below were tested using only the access points that were already installed in the William Gates building for the sole purpose of providing wireless connectivity.

WiFi-based positioning algorithms can be divided into those based on propagation models and those based on radio maps. In the latter, a radio map is constructed during an offline phase prior to the system's deployment, in which beacon signal strengths are recorded at positions throughout the environment. A user is positioned by comparing the results of a WiFi scan performed at the user's current location to the results stored in the radio map. This approach has been shown to outperform model-based algorithms [10], however it has the major drawback that a radio map must be constructed before the system can be used. Until now this task has been done manually. To do this a human stands at different locations in the building, manually marking his position on a map and then performing a beacon scan at each location. This method is very time consuming and is also prone to error, since the user must estimate their true position each time a scan is performed. Furthermore, the map would have to be manually updated if new radio beacons were added, or if existing beacons were moved to new locations. As a result many deployments have been small [12], or restricted to the corridors of larger buildings [10]. Castro et al. describe map-building as "tedious" [34]. In one of the few large-scale deployments of an RF-based location system, 28 man-hours were required to construct a radio map covering a $12,000\,\mathrm{m}^2$ building [35].

One solution to the problem of building radio maps is to use a robot capable of positioning itself within the environment (e.g. using the robot localisation techniques discussed in Section 2.5) to build one automatically. This approach has been used to acquire radio maps for use in robot positioning systems [94]. Crowd-sourcing radio maps has also been proposed [95]. An alternative approach that is made possible by the work in this thesis is to automatically build a radio map whilst a user is tracked by the localisation and tracking system developed in Chapters 5 and 6.

7.1.1 Automatic radio map construction

Radio maps of WiFi access points can be constructed quickly and easily by a user who is being tracked by the localisation system described in Chapters 5 and 6. All that is required is for the user to be equipped with WiFi hardware that is capable of scanning for visible access

points and obtaining corresponding received signal strength indication (RSSI) measurements. Such hardware is provided in a wide range of mobile devices, including the hip-mounted UMPC that forms part of the prototype system described in this thesis. Whilst the user is tracked the UMPC's WiFi hardware is used to repeatedly scan for visible access points and their corresponding RSSIs. When the particle filter updates the state, the most recent WiFi scan to have been completed since the last update (if one exists) is incorporated into the radio map at the user's estimated location. Note that radio map construction does not start until after the user has been localised. If environmental symmetry or the size of the environment prevents localisation, then the user who is building the radio map may manually select their approximate initial location so that the radio map can be built.

Depending on the WiFi hardware used it may be necessary to take special care to ensure that the result of each scan does not contain cached results from any previous scans. Such caching is implemented as a feature by some WiFi device drivers including the MadWifi[1] driver for Atheros chipsets such as the one in the test UMPC. The purpose of such caching is to avoid having to perform a full scan whenever a list of visible access points is requested, since this requires scanning multiple channels and therefore temporarily breaks connectivity to any network to which the device is currently associated. For the tests below a modified version of the MadWifi driver was developed in which the results of previous scans were not cached[2].

The radio map data-structure (and the amount of data stored within it) can be changed according to the desired application. The approach used here divides the 2.5-dimensional map that is used by the localisation filter into cells, which are regions of space within which radio measurements are grouped. A specified cell size determines the granularity of the radio map. For each cell a histogram of measured RSSI values is maintained for each access point that has, at some point, been visible from the cell. A new scan is added to a cell by updating the histogram for each access point that was sighted in the scan. If an access point is sighted for the first time in a cell then a new histogram is created.

Two factors should be considered when deciding how to divide a 2.5-dimensional map into cells:

1. A radio map is most useful for positioning when the variability of radio measurements *within* cells is minimised, and variability *between* cells is maximised. Walls attenuate radio signals far more than air, therefore it is sensible to ensure that cells do not span multiple rooms.

2. To get a complete radio map it is necessary to complete WiFi scans in every cell. Some

[1] http://madwifi-project.org

[2] The necessary driver modifications were implemented by Ripduman Sohan, to whom the author wishes to express his thanks.

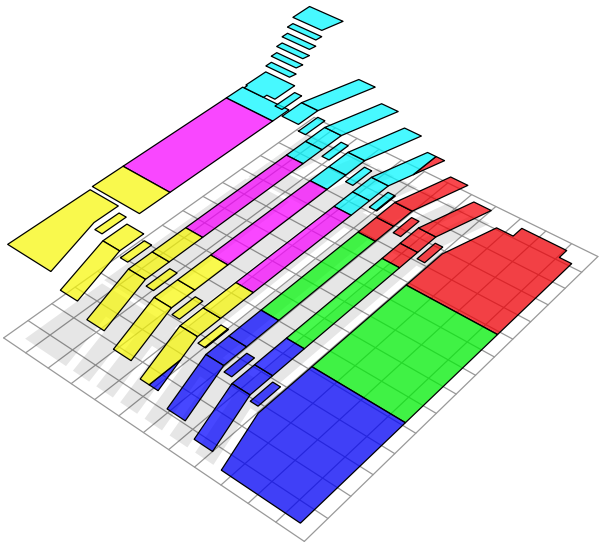


Figure 7.2: The cells generated for a lecture theatre using a cell granularity of $6\,\mathrm{m}$ (each cell is shaded with a different colour). Grid size $1\,\mathrm{m}^2$.

rooms such as the lecture theatre shown in Figure 7.2 are made up of many small floor polygons in the 2.5-dimensional map data-structure (a description of this data-structure can be found in Section 5.1). Therefore it should be possible for cells to span multiple floor polygons within a room.

With these factors in mind, a suitable strategy is to subdivide each room in a 2.5-dimensional map as follows:

1. Compute the 3-dimensional bounding box of each room.
2. Divide the bounding box into the minimum number of identical cuboids such that no edge is longer than the specified cell granularity.
3. A cell is defined as the fragments of floor polygon that are contained within a single cuboid.

The Weiler-Atherton clipping algorithm [96] can be used to compute the fragments of each floor polygon that are contained by each cell, as shown in Figure 7.2.

To construct a complete radio map it is necessary for a user to walk throughout the building that is being mapped. In tests conducted in the William Gates building this required 2 hours and 28 minutes (split into three sessions) in which the user travelled a total distance of $8.7\,\mathrm{km}$, as shown in Figure 7.3(a). Unfortunately access could not be obtained to a number of restricted areas (e.g. plant and server rooms), which are shaded red. In total 40 access points were observed, although seven of these were actually located in neighbouring buildings and were only sighted from a few cells. The radio map constructed for one access point located within the William Gates building is shown in Figure 7.3(b). The overall coverage of the radio map was 99% when using a cell granularity of $5\,\mathrm{m}$, as shown in Table 7.1. In this context cov-

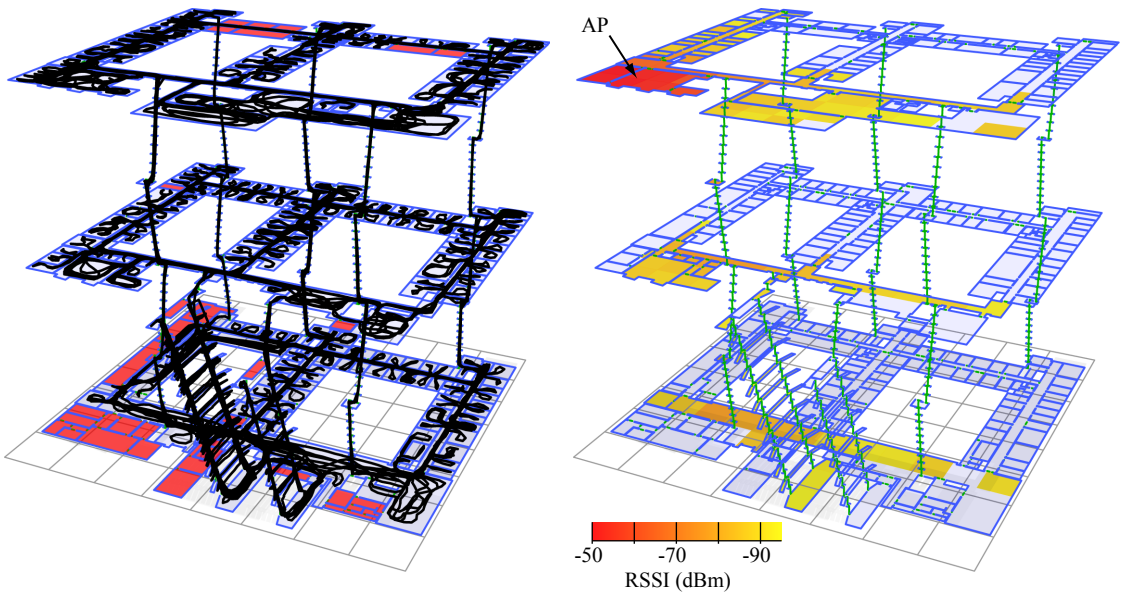


Figure 7.3: Building a radio map. (a) Tracks calculated by the localisation system during radio map construction. Shaded (red) areas indicate rooms that could not be accessed. (b) The radio map generated for a single access point. Cell colour indicates the mean RSSI value.

Table 7.1: The map coverage and mean scans per cell at different cell granularities (excluding restricted areas).

Cell granularity (m)	Coverage (%)	Mean scans per cell
1	59	1.3
2	85	4.3
3	96	9.14
4	99	12.9
5	99	18.9

erage is defined as the percentage of the total floor area (excluding restricted areas) covered by cells in which at least one scan was completed. In a more complete system the user could have been automatically directed to areas that had yet to be mapped whilst moving through the building (e.g. by an application running on a mobile device), however in this case the radio map was completed by manually scanning for visible access points from the few remaining cells.

There are two stages at which WiFi scans together with a radio map can be used to reduce uncertainty during localisation. The first is when the initial distribution of particles (the prior) is generated. The second is during the localisation process itself, in which measurements can be used to weight particles during the filter's correction step. One way in which WiFi scans can be used in both of these stages is through the generation of coarse containment regions within which the user is almost certainly located. The generation and application of such

regions is described below.

7.1.2 Prior generation

During initialisation the WiFi hardware is used to scan for visible access points and obtain corresponding RSSI measurements. Let the vector of observed RSSI values be $\boldsymbol{z}_0 = (\mathrm{ap}_1, \mathrm{ap}_2, ..., \mathrm{ap}_n)$, where ap_n is the RSSI observed for access point n. An initial containment region R_{prior} is then constructed as follows.

1. Calculate the normalised Euclidean distance $\mathrm{ndist}(\boldsymbol{z}_0, \boldsymbol{c}_i)$ between $\boldsymbol{z}_0$ and the vector $\boldsymbol{c}_i$ of mean RSSI values for each cell (indexed by i) in the radio map. The calculation of such distances is discussed below.

2. Construct a prior region consisting of all cells for which the calculated distance is under a threshold $\epsilon = 9.5\,\mathrm{dBm}$. This threshold was determined experimentally (see Section 7.1.4) to ensure that the user's true position is almost certainly contained by the region. Formally the prior region is given by

$$R_{\mathrm{prior}} = \bigcup_i \mathrm{Cell}_i \mid \mathrm{ndist}(\boldsymbol{z}_0, \boldsymbol{c}_i) < \epsilon \tag{7.1}$$

   where Cell_i is the set of polygon fragments that make up the i^{th} cell.

During assisted localisation the initial set of particles is generated uniformly across R_{prior} rather than across the entire floor surface of the building (as is the case in unassisted localisation). The particle headings are distributed uniformly in all directions. KLD-adaptation is used to determine when a sufficient number of particles have been generated, as described in Section 6.1.2. The number of particles that are generated by this approach is (on average) proportional to the total area of the polygon fragments that make up R_{prior}. Hence the number of particles in the prior is proportional to the size of the containment region rather than to the size of the building.

To construct R_{prior} it is necessary to calculate distances between the vector of observed RSSI values and vectors of mean RSSI values stored in the radio map. When calculating such distances it should not be assumed that the same set of access points will be visible throughout the building in which the localisation system is deployed. In fact this is usually not the case in buildings that are large enough such that the use of WiFi could significantly reduce the cost of localisation. Hence the calculation used to obtain $\mathrm{ndist}(\boldsymbol{z}_0, \boldsymbol{c}_i)$ values must give sensible results when $\boldsymbol{z}_0$ and $\boldsymbol{c}_i$ are of different lengths and contain measurements corresponding to different access points. This is a problem rarely addressed in existing literature, since most experimental deployments of WiFi-based location systems have been small and have implic-

itly assumed that all access points are visible from all locations in the environment. Consider the following scan and cell vectors, in which a question mark indicates that an access point was not observed and a more negative value indicates a weaker signal:

$$\boldsymbol{z}_0 = (-82, -73, ?) \tag{7.2}$$

$$\boldsymbol{c}_1 = (-84, -75, ?) \tag{7.3}$$

$$\boldsymbol{c}_2 = (-84, -75, -65)\,. \tag{7.4}$$

One approach might be to compute the distance between $\boldsymbol{z}_0$ and a cell using only the access points for which measurements are present in both vectors. This approach computes equal distances to both cells, yet intuitively the distance to $\boldsymbol{c}_1$ should be smaller, since an access point that is strongly visible in $\boldsymbol{c}_2$ was not observed from the user's location. A better solution is to compute the Euclidian distance over all access points, filling in missing readings in both the cell and the measurement vectors with the weakest RSSI value that can be measured by the WiFi hardware ($-96\,\mathrm{dBm}$ for the Atheros chipset in the OQO UMPC). This approach yields a larger distance to $\boldsymbol{c}_2$ as required. Finally, each distance measurement is normalised by dividing by the number of access points that are visible from the cell according to the radio map. This makes it possible to compare distance measurements to cells from which different numbers of access points are visible. Note that this penalises a missing value in the cell vector more than a missing value in the measurement vector. For example consider the following vectors and the normalised distance between them:

$$\boldsymbol{z}_{0\mathrm{a}} = (-82, -73, ?) \tag{7.5}$$

$$\boldsymbol{c}_{1\mathrm{a}} = (-84, -75, -75) \tag{7.6}$$

$$\mathrm{ndist}(\boldsymbol{z}_{0\mathrm{a}}, \boldsymbol{c}_{1\mathrm{a}}) = 7.1\,\mathrm{dBm}\,. \tag{7.7}$$

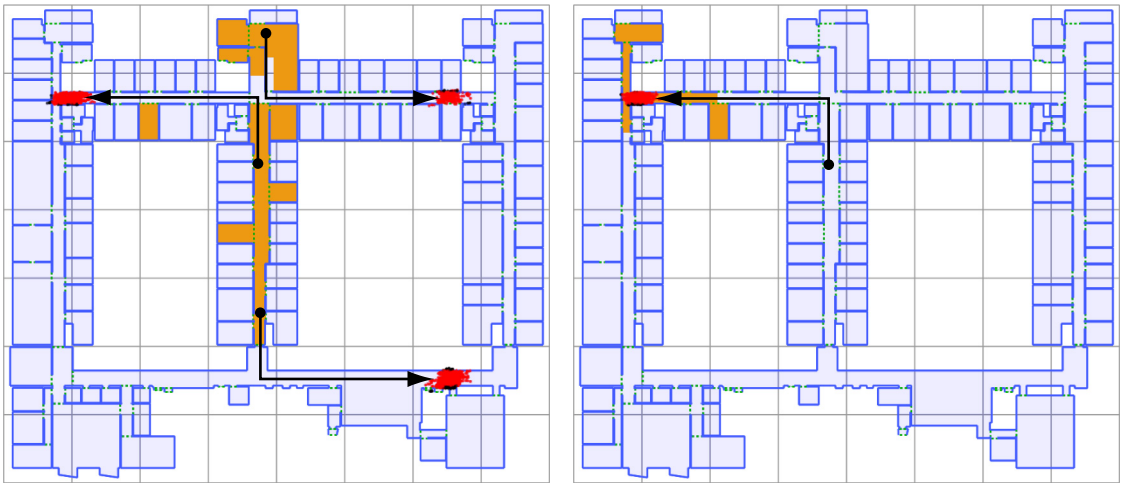
If the cell and measurement vectors are switched then a larger distance is obtained:

$$\boldsymbol{z}_{0\mathrm{b}} = (-84, -75, -75) \tag{7.8}$$

$$\boldsymbol{c}_{1\mathrm{b}} = (-82, -73, ?) \tag{7.9}$$

$$\mathrm{ndist}(\boldsymbol{z}_{0\mathrm{b}}, \boldsymbol{c}_{1\mathrm{b}}) = 10.6\,\mathrm{dBm}\,. \tag{7.10}$$

This is actually a desirable property, since each cell vector is typically derived from many measurements (i.e. all of the scans that were performed from within the cell during radio map construction). If an access point were visible from a cell then you would expect that it would have been sighted in at least one of them. In contrast the measurement vector $\boldsymbol{z}_0$ corresponds to a single scan during which an access point that it is possible to sight from the user's location may have been obscured by the user's body or some other object in the environment.



(a) (b)

Figure 7.4: The result of applying the localisation filter to the same example as in Figure 7.1 when (a) R_{prior} (indicated by the shaded region) is used. (b) When both R_{prior} and containment measurements are used (the shaded area indicates the most recently applied measurement). An arrow indicates the path followed by each cluster.

Figure 7.4(a) shows the result of using R_{prior} in the same example as shown in Figure 7.1. Note that the prior region covers only a small proportion of the total floor area. Since the number of particles is proportional to the floor area that they cover, the computational cost of localisation has been reduced. By using the prior the number of distinct clusters is reduced from twelve to three, since only three of the clusters shown in Figure 7.1 originate from inside the prior containment region. Hence uncertainty due to symmetry in the environment has also been reduced. Despite this the use of the R_{prior} was not sufficient to unambiguously determine the user's true location. In particular the use of a prior containment region is not sufficient to resolve ambiguity between multiple clusters that have different orientations but similar initial positions (that are all inside the region). Such clusters often arise due to rotational symmetry in the environment.

7.1.3 Containment measurements

To remove clusters that arise due to rotational symmetry it is possible to generate and apply containment regions throughout the localisation process. For a filter update at time t a suitable containment region is given by

$$R_t = \bigcup_i \text{Cell}_i \mid \text{ndist}(\boldsymbol{z}_t, \boldsymbol{c}_i) < \epsilon \tag{7.11}$$

where $\boldsymbol{z}_t$ is the result of the most recently completed WiFi scan since the previous update. The region is then applied as a *containment measurement* during the correction step of the

particle filter's update algorithm. The containment measurement corresponding to the region R_t is given by

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t) = \begin{cases} 1 & \text{if } \boldsymbol{x}_t \text{ is in } R_t \\ 0 & \text{otherwise} \end{cases} \tag{7.12}$$

where $\boldsymbol{x}_t$ is the state of the particle whose weight is being corrected. This sets the weight of all particles outside of the containment region to zero, causing them to be removed during re-sampling. Figure 7.4(b) shows the result of using both R_{prior} and containment measurements during localisation. In this example only a single cluster of particles remains, whose position corresponds to the user's true location.

7.1.4 Robustness

In the algorithm presented above WiFi signal strength vectors are used to generate containment regions within which the probability $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ is assumed to be constant. An alternative (and more conventional) approach would be to compute a probability $p(\boldsymbol{z}_t|\boldsymbol{c}_i)$ for each cell and then apply a measurement with the probability distribution

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t) = p(\boldsymbol{z}_t|\boldsymbol{c}_i) \qquad \boldsymbol{x}_t \text{ is in } \mathrm{Cell}_i\,. \tag{7.13}$$

For each cell the probability distribution $p(\boldsymbol{z}_t|\boldsymbol{c}_i)$ could be calculated directly using the signal strength histograms stored in the radio map. The main advantage of this approach is that particles can be assigned weights of any value from 0 to 1, whereas containment measurements can assign weights of 0 and 1 but nothing in-between. This finer granularity makes it possible for surviving particles to be weighted based on how consistent they are with measurements received over multiple updates.

Unfortunately this approach also suffers from a major problem: that the distributions $p(\boldsymbol{z}_t|\boldsymbol{c}_i)$ are not typically constant in a real environment. This is because received signal strengths at a particular location are affected by changes in the local environment such as the opening or closing of a door [33]. Such a change will introduce a change in the corresponding probability distribution that will not be reflected in a pre-constructed radio map. The result is a systematic error in the radio map that can draw the particle cloud away from the user's true position. In the worst case this could cause the entire cloud to violate an environmental constraint, resulting in a localisation failure. Containment regions avoid this problem by assuming that the distribution is uniform over the entire region. This avoids introducing a bias either towards the user's true position (when the radio map is correct) or away from it (when it is not).

One problem that can arise when applying containment regions is that the cluster corresponding to the user's true position might fall outside of the region and be removed. To ensure that this is unlikely to occur the threshold ϵ must be set to a sufficiently large value. In particu-

lar the threshold should be large enough such that a containment region will still contain the true position even when systematic errors are present in the radio map due to small changes in the environment. A suitable threshold was established for the William Gates building by processing further data obtained using the prototype system that was distinct from the data that was used to construct the radio map. The new data consisted of walks lasting a total duration of just under one hour that were performed a week after the radio map was built (during which small changes in the environment such as the opening and closing of doors would have occurred). RSSI measurements obtained during the walks were not used to actively aid the localisation process, but were instead passively compared to the radio map during tracking. After each step during tracking the normalised distance $\mathrm{ndist}(\boldsymbol{z}_t, \boldsymbol{c}_i)$ was calculated to each cell that overlapped the single cluster of particles calculated by the filter. The cumulative distribution of the calculated distances is shown in Figure 7.5. The probability of the user being located outside of a containment region can be estimated from this distribution. For example if a containment region is generated using a threshold of $\epsilon = 3.94\,\mathrm{dBm}$ then the probability that it does not contain the user's true position can be estimated as 0.5. For a threshold of $\epsilon = 9.5\,\mathrm{dBm}$ this probability falls to 0.00028. When using this value no localisations were found to require more than 74 steps. Hence for the worst case in which 74 steps are required and a containment measurement is applied during each update, the overall probability of a failure during localisation can be estimated[3] as

$$1 - (1 - 0.00028)^{74} = 0.021\,. \tag{7.14}$$

In an average case requiring 15 steps the probability of failure is just 0.0042. This failure rate was deemed acceptable and hence a threshold of $\epsilon = 9.5\,\mathrm{dBm}$ was selected for use during evaluation. When a failure occurs the localisation process can simply be restarted from the user's current location. Note that different thresholds may be required for different buildings, access point topologies and WiFi hardware.

7.1.5 Localisation performance

In order to quantify the benefit of assisted localisation using WiFi RSSI measurements, a further dataset was obtained as a user walked continuously for 35 minutes throughout the William Gates building. The localisation filter was applied repeatedly to the data from a number of randomly selected starting points. In total 50 random starting points were drawn uniformly from the first 30 minutes of the walk. Starting points were not drawn from the final five minutes in order to ensure that there was always sufficient remaining data to allow the user to be localised. From each starting point the filter was applied three times: once without

[3]It is assumed that the probability of failure is independent for each containment region. This is almost certainly not the case, however it is assumed in order to obtain an indicative measure.

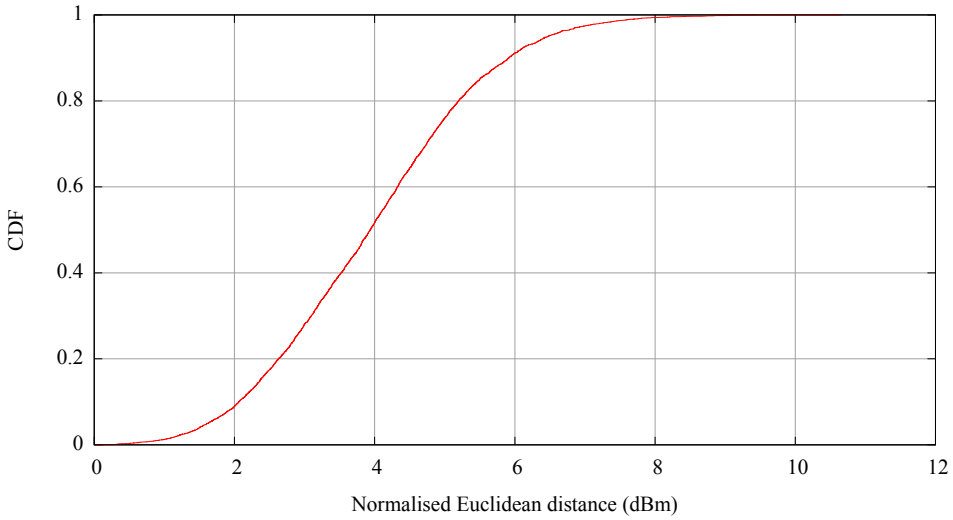
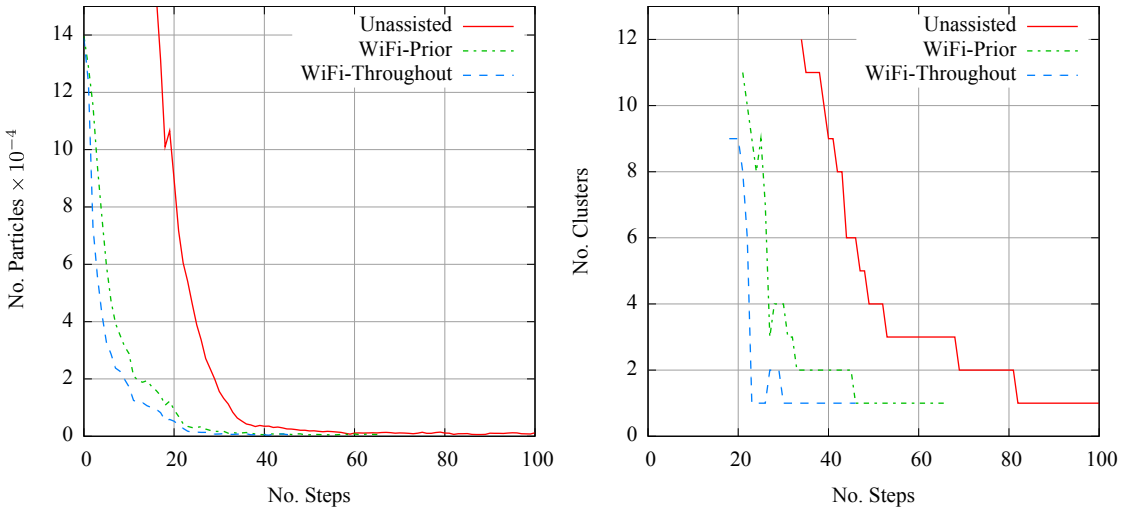


Figure 7.5: The cumulative distribution of $\mathrm{ndist}(\boldsymbol{z}_t, \boldsymbol{c}_i)$ for cells overlapped by a cluster of particles corresponding to the user's true location.



(a) (b)

Figure 7.6: The data gathered for a single localisation without using the radio map, using R_{prior} only, and using both R_{prior} and containment measurements. (a) The number of particles generated at each step. (b) The number of clusters at each step (clustering is only performed when the number of particles is below $8,000$).

Table 7.2: A summary of WiFi-assisted localisation performance in the William Gates building (excluding failed runs).

	Radio map use	Mean	Std-dev	Max	Min
Particles in prior	None	$1,271,300$	$9,124$	$1,280,781$	$1,252,106$
	Prior only	$191,528$	$118,025$	$459,514$	$45,649$
	Throughout	$191,457$	$117,951$	$459,886$	$45,963$
Steps until localisation	None	71	27.8	146	24
	Prior only	49	24.3	140	9
	Throughout	34	14.4	74	7

using the radio map, once using R_{prior} to constrain the user's initial position and once using both R_{prior} and containment measurements throughout the localisation process. Particle and cluster counts were recorded throughout each run.

The results obtained for a single starting point are shown in Figure 7.6. In this example the prior contained $1,277,890$ particles and 82 steps were required to localise the user without WiFi assistance. When R_{prior} was used to constrain the initial location of the user $139,626$ particles were generated and 46 steps were required to determine the user's location. When containment measurements were used throughout only 30 steps were required.

Table 7.2 summarises the results that were obtained[4]. These results exclude one failed run in each scenario and an additional failed run when containment measurements were used throughout. One starting position was responsible for a failure in each of the scenarios. This case was a particular difficult one in which the user passed at a tight angle through a narrow doorway early during the localisation process. For this unusual case the number of particles determined by the adaptive scheme was not sufficient. The additional failure when containment measurements were used throughout was caused by an incorrect containment measurement (i.e. one whose containment region did not contain the user's true position).

As expected the use of R_{prior} significantly reduced the number of particles in the priors. There was however a large variation between the number of particles generated in different runs, with runs from one starting point generating in the order of $460,000$ particles. This is explained by the presence of large open plan areas in the William Gates building, within which radio signals propagate far more freely relative to propagation through areas containing many walls. When the user is located in such an area the containment region obtained using WiFi signal strengths is likely to be large, resulting in many particles. The average number of steps taken by the user before localisation was also reduced by using the constrained prior, however the worst case of 146 steps was not significantly reduced. Hence R_{prior} alone was not always sufficient to reduce the problem of environmental symmetry.

[4]The particle counts given here are significantly lower than those presented in previous work [97]. This is because the results in [97] were obtained using a version of the localisation filter in which KLD-adaptation parameters had not been tuned to use as few particles as possible (see Section 6.1.3).

Using containment measurements in addition to R_{prior} reduced the average number of steps required to locate the user to 34. The worst case was reduced to 74 steps. This demonstrates that the use of containment measurements helps to resolve ambiguities that arise due to symmetry in the environment. This is achieved at the cost of the additional computational overhead required to generate and apply such measurements.

The cost of generating a containment measurement is $\mathcal{O}(A)$ where A is the floor area of the building, since generating a containment region requires the calculation of a distance to each cell in the radio map. Fortunately these computations are very cheap. For the William Gates building 99% of the generated containment regions were constructed in under $5\,\text{ms}$ in a Java implementation running on a $2.6\,\text{GHz}$ desktop PC, which is negligible relative to the time required to update the numbers of particles that are used during localisation (see Section 5.3.2). The weight of a particle is updated during the correction step by first finding the cell in the radio map within which it is located and then multiplying its weight by 0 or 1 depending on whether that cell is part of the containment region. This requires $\mathcal{O}(1)$ time[5] and in practice the overhead is negligible relative to the time required to update the particle's state. Hence the overhead of generating and applying containment regions is small even for large buildings and is nearly always worthwhile.

7.2 Mapping other signals

The techniques developed for WiFi-assisted localisation in Section 7.1 can be applied to other radio signals. The only requirement is that the measurable signal properties (e.g. RSSI in the case of WiFi) must be both sufficiently stationary (meaning that similar measurements will be observed from a given location at different points in time) and vary sufficiently across the building so that they can be used to constrain the user's location.

The offline construction and subsequent use of maps to assist the localisation filter is not only applicable to radio signals. In fact it is possible to construct a map of any property that can be sensed by devices that are attached to the user. Such a map can then be used to assist during localisation provided that the sensed property satisfies the two requirements mentioned above. One particularly interesting example is to map magnetic disturbances throughout the environment, since many commercial IMUs already contain magnetometers that are able to sense such disturbances. It was established in Section 4.3 that the magnitude and frequency of such disturbances make it impossible to determine the user's heading in the William Gates building using the magnetometers in a foot-mounted IMU. This property is actually beneficial when using a pre-constructed map of magnetic disturbances, since it means that there will be many significant features in the map that can be compared to those currently sensed at the

[5]The radio cell can be determined in $\mathcal{O}(1)$ time due to the reference poly_t in the state vector of the particle.

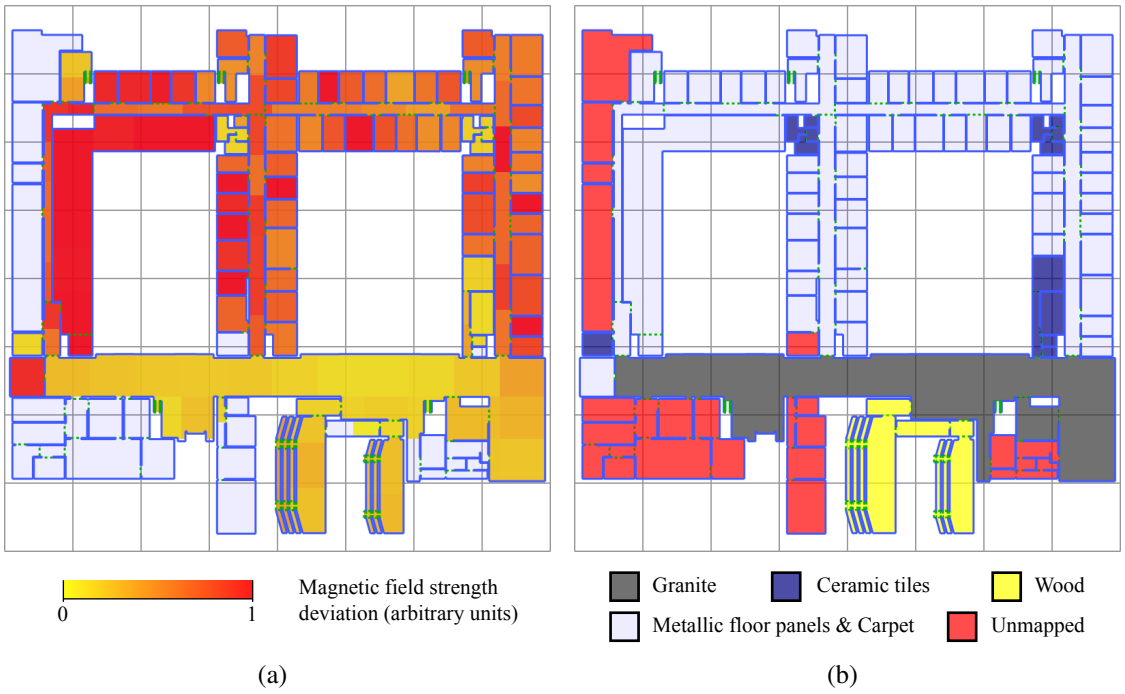


Figure 7.7: Mapping magnetic field deviations on the ground floor of the William Gates building. (a) A map of the mean within-step deviation in magnetic field strength, scaled such that the maximum calculated value is equal to one. (b) Different floor surface materials in the building.

user's location. Furthermore the sources of such features are often static, for example the metallic floor panels installed throughout large portions of the William Gates building.

Figure 7.7 demonstrates a very simple approach to mapping deviations in magnetic field strength throughout the ground floor of the William Gates building. During each step the magnetic field strength was sampled at 100 Hz using the magnetometers contained within the foot-mounted IMU. At the end of a step the standard deviation of the magnetic field strengths recorded during that step was calculated and inserted into the map. A large deviation indicates that the user's foot passed through a significant magnetic disturbance during the step. Figure 7.7(a) shows the mean deviations recorded in each cell of the map. It is interesting to note that there is a correlation between the mapped deviations and the use of different floor surface materials in different parts of the building, as shown in Figure 7.7(b). Large deviations were recorded in areas where metallic floor panels were used, whereas far smaller deviations were recorded in parts of the building containing stone, wood and ceramic floor surfaces. Thus even this simple approach may be sufficient to determine whether the user is currently in a region of the building in which metallic floor panels are installed. A more sophisticated approach may allow the user to be further constrained, for example by mapping the direction of the magnetic field throughout the building. The development of such approaches and the application of magnetic field maps during localisation is left as future work.

7.3 Assisted tracking

Although this chapter has focused on the use of additional measurements during localisation, such measurements can also be applied during tracking. For example containment measurements generated from WiFi scans can be applied throughout the tracking phase as well as during the initial localisation phase. Although this is possible, it is rarely beneficial. For example WiFi containment measurements will not improve the accuracy of the positions calculated during tracking due to their coarse size and uniform nature. Furthermore a tracking failure may occur whenever a containment measurement is applied, since there is a small but finite chance that the user's true position will not be contained by the corresponding containment region. Hence the application of containment measurements during tracking is actually counter-productive.

In some cases additional measurements can be applied in order to improve accuracy during tracking. For example position measurements from the Active Bat system could be used whenever a user passes through the wing of the William Gates building in which it is deployed. Although this is possible, it usually makes more sense to use entirely different tracking algorithms when measurements from relatively accurate positioning systems are available. In this example an INS together with the error-state Kalman filter that was developed as a ground truth in Section 3.4 can be used to track the user's position more efficiently, more accurately and at a much higher frequency when in the area of the building that is covered by the Active Bat system. When the user exits this area the localisation filter could then resume tracking, with an initial set of particles that are clustered around the last position to be calculated by the INS.

7.4 Conclusions

A wide range of approaches can be used to assist the process of global localisation. All such approaches require some form of additional information to be provided to the localisation filter that can be used to reduce uncertainty in the user's state. There are a range of sources from which such information can be obtained including manual input, additional on-body sensors and measurements obtained from sensors in the environment. The most suitable approach is application dependent. When it is not possible to access a building in advance the user could be asked to provide their approximate position at the start of localisation. When prior access is available it is often possible to map signals that propagate through the environment, such as those emitted by WiFi access points. One of the major advantages of the localisation system developed in this thesis is that it can be used to construct such maps automatically whilst the user is tracked moving through the environment.

Chapter 8

Conclusions

8.1 Research contributions

This thesis has demonstrated that it is possible to track the absolute position of a pedestrian in an indoor environment without relying on any fixed infrastructure. This has been achieved by combining pedestrian dead reckoning (PDR) techniques with ideas from the related field of robot localisation.

The pedestrian localisation and tracking system that has been developed in this thesis consists of a micro-machined electromechanical systems (MEMS) inertial measurement unit (IMU) and two Bayesian filters. The first of these is a PDR filter, consisting of an error-state Kalman filter and an inertial navigation system (INS). This filter processes measurements obtained from the IMU in order to track the relative movement of the user. Drift errors that would normally accumulate rapidly in the tracked position are greatly reduced by the application of zero velocity updates (ZVUs). This approach is made possible by mounting the IMU on the user's foot, which is known to be stationary whenever it is grounded. The PDR filter segments the user's relative movement into step events that are passed to a second Bayesian filter.

The second Bayesian filter is a localisation particle filter that combines step events calculated by the PDR filter with a map of the environment. A 2.5-dimensional map data-structure was developed to provide efficient support for multi-storey buildings, which were not supported by existing robot localisation algorithms. By enforcing environmental constraints that are defined by a 2.5-dimensional map, the localisation filter is able to narrow down the position of a pedestrian as they move through the environment. Once the particles have converged to form a single cluster around the user's position, the same filter is then able to track the user's absolute position without becoming less accurate over time.

A number of techniques have been presented in this thesis that improve the performance of the

basic localisation and tracking system. The most important of these is the dynamic adaptation of the number of particles used by the localisation filter based on the amount of uncertainty in the user's position. This has the effect of automatically reducing the computational requirements of the filter as the user's position becomes more certain. The use of furniture constraints has also been shown to improve tracking accuracy in open plan environments. Finally, additional sensors can be used to reduce the localisation filter's computational requirements and improve the speed with which it can determine the user's position.

The purpose of developing the pedestrian localisation and tracking system outlined above was to answer the research questions posed in Section 1.1. These questions are repeated below and are addressed in the following sections, which also summarise the main contributions and conclusions of this thesis.

1. How rapidly does drift accumulate in an inertial navigation system when using current state-of-the-art MEMS inertial sensors?
2. How accurately can the relative movement of a pedestrian be measured using MEMS inertial sensors?
3. What is the minimum infrastructure or knowledge required to determine a pedestrian's absolute location and heading?
4. Can environmental constraints be used to prevent the accumulation of drift over time?

8.1.1 Research question 1: Unconstrained inertial navigation

One of the most important conclusions of this work is that unconstrained inertial navigation is an unsuitable technique for tracking the relative movements of a pedestrian given current inertial sensor technologies. Chapter 3 showed that with a current state-of-the-art MEMS IMU, drift accumulates in the estimated position that typically exceeds $170\,\mathrm{m}$ after one minute of tracking. Such drift arises due to measurement errors incurred by each of the individual accelerometers and gyroscopes that are contained within an IMU. These individual errors propagate through the inertial navigation system equations, causing drift in position that grows proportional to t^2 in the short term (when the accelerometers are the primary cause of drift) and t^3 in the long term (when the gyroscopes are the dominant error source), where t is the period of time for which the user is tracked. This rapid error growth means that even big improvements in the quality of MEMS inertial sensors will only result in modest increases in the duration for which a user can be tracked before the error in position exceeds a given threshold.

Given that many indoor location-aware applications require sub-metre accuracy (so that they can determine whether a person is in close enough proximity to another object for a physical

interaction to take place) for extended periods of time, it is expected that the use of unconstrained inertial navigation will remain infeasible for indoor pedestrian tracking for the foreseeable future. If it is to become feasible it will be necessary not only to develop IMUs that are many orders of magnitude more accurate than those of today, but also new technologies for modelling and compensating for local gravitational variations.

8.1.2 Research question 2: Constrained pedestrian dead reckoning

Due to the limitations of unconstrained inertial navigation it is necessary to seek additional constraints or external measurements that can be used to correct an INS that is used to track people or objects in an indoor environment. In Chapter 3 a general purpose error-state Kalman filter was developed for enforcing such constraints. This approach is ideal due to the Kalman filter's computational efficiency and its ability to track correlations between growing errors in different components of the state of an INS. By tracking such correlations the filter is able to correct components of the state that are not directly observed by measurements. For example it is able to correct errors in tilt, velocity and displacement when a measurement of the IMU's velocity is received. More generally Bayesian filters (of which the Kalman filter is a specific type) allow uncertainty in each measurement to be taken into account. Within the Bayesian filter framework a constraint can be enforced by generating a *pseudo-measurement*; a fake measurement that specifies the nature of the constraint.

In Chapter 4 an INS was used together with the error-state filter to construct a PDR filter capable of tracking the relative movements of a pedestrian equipped with a foot-mounted MEMS IMU. This sensor placement allowed the application of ZVU pseudo-measurements whenever the user's foot was known to be in contact with the floor. In one example it was shown that by using this approach a pedestrian's relative movements could be tracked to within $3.97\,\mathrm{m}$ over a period of just under nine minutes, compared to an error of $43.15\,\mathrm{km}$ when using unconstrained inertial navigation.

It is important to note that the PDR filter is more specialised than general purpose inertial navigation algorithms. In particular it requires the user to be walking and the IMU to be mounted on the user's foot. As the quality of IMUs continues to improve it may be possible to make use of more generic constraints and thus develop less specialised algorithms. Such algorithms may be more generally applicable, tracking arbitrary objects rather than being restricted to pedestrians. Furthermore such algorithms will still be suitable for providing relative movement events to a localisation particle filter such as the one developed in Chapter 5.

8.1.3 Research question 3: Determining absolute location and heading

The third research question posed at the beginning of this thesis asked what minimal infrastructure or knowledge was required to determine the absolute location and heading of a person in an indoor environment. Determining this state is necessary if relative movement measurements obtained from a PDR filter are to be used to track a user's absolute position rather than their position relative to an unknown initial starting point and orientation. Chapter 5 demonstrated that particle filters could be used to infer the absolute location and heading of a person given only relative movement measurements and a map of the building. The map provides constraints such as walls and floors that naturally limit the movement of a person in an indoor environment. By enforcing these constraints a particle filter is able to narrow down the location of a user as they move through the environment, until their position can be uniquely determined. This is known as pedestrian localisation.

The localisation algorithms developed in Chapters 5 and 6 demonstrate that it is possible to determine a user's absolute location and heading without the need for any fixed infrastructure, however there are a number of caveats. Firstly the user is required to move in order for their location to be narrowed down. Hence it is impossible to locate a stationary pedestrian using this approach. Secondly the localisation process is computationally expensive, since a large number of particles are required to represent all of the possible positions and orientations of the user. Thirdly environmental symmetry exhibited by the environment may delay or prevent the user's state from being determined. This is particularly problematic for high rise buildings in which many of the floors have very similar layouts.

To address the problems outlined above the user's initial state can be approximated (or at least bounded to some region of space) using additional measurements. This is known as assisted localisation. Possible approaches for approximating the user's initial location include the use of sparsely deployed beacons such as WiFi access points, using the last location estimate calculated by an outdoor positioning system as a user moves into an indoor environment and asking the user to enter their approximate position manually. Chapter 7 showed that by using WiFi measurements in the William Gates building the average number of particles required during the first filter update could be reduced from $1,271,300$ to $191,500$, thus reducing both the computational and memory requirements of the localisation process. The average number of steps taken by the user before their position was determined was also reduced from 71 to 34.

8.1.4 Research question 4: Use of environmental constraints

The main contribution of this work has been to demonstrate that it is possible to use environmental constraints to prevent the accumulation of drift that would otherwise occur when

tracking a pedestrian using inertial navigation techniques. The approach demonstrated in Chapters 5 and 6 used a particle filter to continually enforce constraints such as walls and floors in order to maintain a drift free estimate of the user's state. The computational and memory costs are low relative to the cost of global localisation, since far fewer particles are required. It was shown that the absolute position of a pedestrian could be tracked indefinitely to within $0.62\,\mathrm{m}$ 95% of the time in a typical office environment. This level of accuracy is sufficient for location-aware applications that need to be able to infer when a user is standing close enough to an object for a physical interaction to take place (e.g. when a user is close enough to a workstation to be using it).

It is important to note that the accuracy of the localisation and tracking system developed in this thesis is dependent on the density of environmental constraints. In areas where there are few constraints the tracking accuracy is worse than in areas where there are many constraints. For open plan offices furniture such as desks and cubicle dividers can be used as additional constraints in order to improve tracking performance. OpenRoomMap was developed as a way of crowd-sourcing a world model of such constraints in the William Gates building. When using the furniture constraints obtained through OpenRoomMap it was possible to position a user to within $0.94\,\mathrm{m}$ 95% of the time in a simulated open plan environment, compared to $1.58\,\mathrm{m}$ 95% of the time when furniture constraints were not used.

8.2 Future work

Pedestrian localisation and tracking in indoor environments is still far from being a solved problem. There are many avenues of future work that must be explored before complete systems based on the algorithms developed in this thesis can be deployed. These avenues can be divided into two broad categories: enhancements to the algorithms themselves and determining how such algorithms can be deployed in real-world systems. Further work that has been identified in both of these areas is outlined below.

8.2.1 Algorithmic enhancements

There are numerous possible enhancements that can be made to the data-structures and algorithms presented in this thesis, ranging from minor performance improvements to major changes that add additional functionality. In the latter case generalisation to support moving floor surfaces such as those found in elevators is of particular interest. One way in which this can be achieved is outlined in Section 8.2.1.1. A list of more minor topics and possible enhancements is given in Section 8.2.1.2.

8.2.1.1 Moving floor surfaces

Large buildings often contain areas in which there are moving floor surfaces on which a pedestrian may stand. Examples of such areas include elevators, escalators and travelators. It is necessary to make a number of modifications to the algorithms developed in this thesis in order to support such cases. One possible approach is outlined below, for which an extremely simple proof of concept implementation has been developed.

PDR filter generalisation The PDR filter developed in Chapter 4 assumes that the user's foot has a velocity of zero whenever it is grounded. This assumption is violated by moving floor surfaces and hence it is necessary to extend such a filter to be able to detect and adapt to situations in which a pedestrian walks onto a moving floor surface. One crude approach could be to look for unexpected changes in velocity. Moving floor surfaces usually have velocities that are far greater than the drift in velocity that can occur during a short period of time. Hence it should be possible to reliably disambiguate between drift errors and cases in which a pedestrian steps onto a moving floor surface. When such a transition occurs zero velocity constraints should be applied by simply resetting the velocity component of the filter to zero (so as to avoid incorrectly correcting other components of the state such as the orientation and displacement), after which the PDR filter will track the user's relative movement in a moving frame of reference corresponding to the floor surface.

Map generalisation The 2.5-dimensional map data-structure must be extended so that it is possible to represent areas in which there are moving floor surfaces. Rather than consisting of connected of floor polygons, an extended 2.5-dimensional map could consist of connected *spatial zones* of different types. For the majority of the building a floor polygon type could be used, with other types used to represent areas containing moving surfaces. For example an elevator zone could be defined by a polygon representing the base of the elevator shaft and a list of heights at which the elevator can stop, each with a connection from an edge of the elevator shaft into another spatial zone at that height.

Localisation filter generalisation Rather than being constrained to lie on a floor polygon, each particle in an extended localisation filter could be constrained to lie within a spatial zone. Different motion models could be applied to the particles in zones of different types to model the movement of floor surfaces. For example particles in an elevator zone could have their vertical displacements periodically updated in order to model the possible movement of the elevator. It may also be necessary to augment the state of each particle in such zones. For example particles in an elevator could have their states extended to model the state of the elevator (i.e. whether it is stationary, going up or going down).

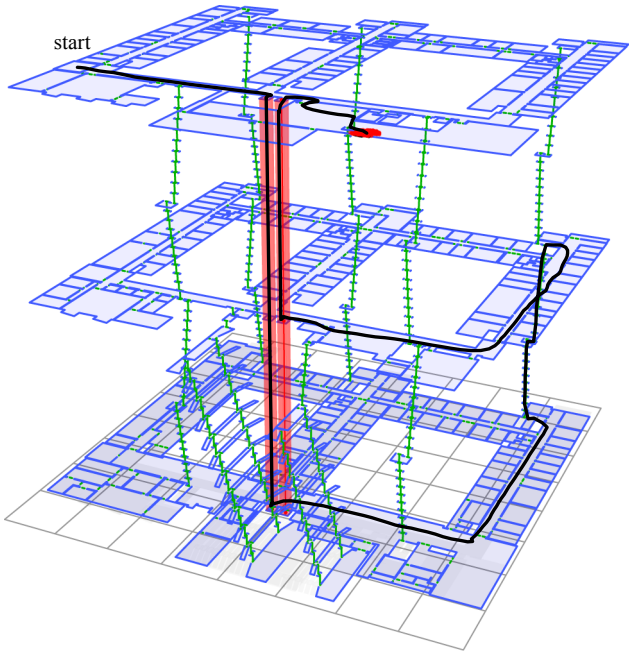


Figure 8.1: An example path calculated using a prototype extended localisation filter. The two elevator shafts are shaded (red). Grid size $= 10\,\mathrm{m}^2$.

Figure 8.1 shows the result of tracking a pedestrian using a prototype version of the localisation and tracking system that was extended to support the elevators in the William Gates building. The user was correctly tracked as he descended two floors in an elevator and later as he ascended one floor to his final position. In this prototype, the PDR filter was modified to detect unusual changes in the IMU's vertical velocity. On detecting such a change, zero velocity constraints were applied simply by resetting the velocity of the underlying INS to zero. At the same time an additional measurement was generated to indicate the change to the localisation filter. The 2.5-dimensional map was extended to include elevator zones as described above. The state of each particle was augmented on entering an elevator zone, with the initial augmented state always set to indicate that the elevator was stationary. Particles had their vertical displacements periodically updated when in the elevator zones based on their augmented states, which were updated based on the additional measurements reported by the PDR filter. Although this is only a crude demonstration, it does at least show that the approach described is a feasible way in which moving floor surfaces can be supported. Far more work is required in the future to further develop the ideas outlined above into a robust solution that can be applied to moving floor surfaces of different types.

8.2.1.2 Additional algorithmic enhancements

There are many other topics of research and possible algorithmic enhancements that could be investigated in the future. Note that the following list is by no means exhaustive.

- It has been established that the performance of the system developed in this thesis is highly dependent on the environment (i.e. the amount of symmetry exhibited by the environment as well as the density of environmental constraints). It will be necessary to evaluate such systems in a large number of different environments in order to investigate these dependencies in more detail and to identify any cases that are particularly problematic.

- Variations on the standard bootstrap particle filter may yield minor performance improvements. For example the clustered particle filter described in Section 6.2.1 is likely to reduce the probability of localisation failures in some environments.

- There are many ways to assist the localisation process in addition to the use of WiFi access points, some of which are described at the start of Chapter 7. Many mobile phones are now equipped with GPS receivers that are able to track the position of users when they are outdoors. Hence it is of particular interest to develop systems that are able to seamlessly transition from GPS to indoor positioning. For example the last position calculated using GPS prior to a user entering a building could be used to determine the entrances into which the user could have walked. A localisation filter could then be initialised with particle clouds generated only in the vicinity of these entrances.

- In addition to GPS receivers, many mobile phones now contain MEMS inertial sensors. Thus it is desirable to develop PDR algorithms that are able to use measurements made by these sensors. Such algorithms should ideally work correctly when user carries their phone in different ways, for example clipped to their belt or in their pocket, hand or bag. Although algorithms have already been developed for this purpose they are significantly less accurate compared to the foot mounted approach. For example the foot mounted approach has been shown to accumulate drift in the order of 0.3% of the total distance travelled (see Section 2.2.2.2), whereas even algorithms that require a relatively stationary mounting point on the user's body (as might be the case when using sensors in a phone clipped to a user's belt) typically incur positioning errors that are closer to 3% of the travelled distance (see Section 2.2.2.1). Attempts to perform dead reckoning using sensors in the pocket have proved even more error prone [48]. The development of better dead reckoning algorithms that do not require sensors to be mounted on the user's foot may be made possible by further improvements in the quality of MEMS inertial sensors.

- Probabilistic constraints can be used to represent uncertainty in the model of the environment. Such an approach may be useful for representing constraints imposed by movable objects such as desks, as described in Section 6.3.3. Further work is needed to investigate the extent to which they can be used. One particularly interesting question is whether it is possible to automatically vary the probability of constraints based on their

interactions with the particle cloud. For example the probability of a constraint could be automatically reduced whenever it is responsible for a tracking failure, since such a failure indicates that the constraint may not be present in the physical environment.

- There are many other data-structures in addition to the 2.5-dimensional map that can be used to represent indoor environments. Of particular interest are graph-based representations such as Voronoi diagrams. Such representations may allow the development of localisation and tracking algorithms that require an order of magnitude less computation, whilst incurring little (if any) loss in tracking accuracy. The use of Voronoi diagrams was discussed in more detail in Section 6.3.4.1.
- Commercial mapping companies are beginning to map indoor environments, however it is unrealistic to expect that such maps will be kept permanently up to date. This poses a problem for the localisation algorithms developed in Chapters 5 and 6, which rely on the availability of an accurate model of the building. A crowd-sourcing approach such as the one used by OpenRoomMap may allow maps to be updated more frequently than is viable for commercial mapping companies. Probabilistic constraints may also offer a partial solution to deal with uncertainty in the map itself. Nevertheless the construction and maintenance of world models in general remains an open research topic that is particularly relevant to pedestrian localisation and tracking.

8.2.2 Real-world deployment

A major question that must be addressed in the future is how algorithms such as the ones developed in this thesis might be deployed in real-world positioning systems. When designing and deploying such systems it is necessary to consider many practical constraints, such as the limited computational resources and battery lives of mobile devices. It is also necessary to consider user concerns such as location privacy and the inconvenience of wearing any device necessary for the system to function. Finally, real-world positioning systems must support multiple users. This can be achieved by executing multiple instances of the algorithms developed in this thesis, however it may be possible to develop algorithms that are optimised for locating and tracking multiple users.

8.2.2.1 Limited power and computational resources

Mobile devices that can be carried by pedestrians are far less computationally powerful than servers and desktop PCs. The electrical power that a mobile device has available is also limited by current battery technology. In a practical implementation of a pedestrian localisation and tracking system it is necessary to power the sensors, the hardware used to execute the

necessary algorithms and in some cases a radio to communicate with other systems (e.g. to publish the user's current position).

Current high end IMUs have power requirements that make them unsuitable for continuous use in mobile devices. For example the Xsens Mtx IMU requires $350\,\mathrm{mW}$. A typical mobile telephone battery is currently rated at $1400\,\mathrm{mAh}$ at $3.7\,\mathrm{V}$[1]. Such a battery would be exhausted after less than 15 hours by the IMU alone. Hence it will be necessary to investigate power management strategies (e.g. turning off the IMU when the user is stationary) as well as developing more power efficient inertial sensors and higher capacity batteries (both of which are current industry trends).

Whilst most absolute positioning algorithms perform a near constant amount of computation per update, localisation algorithms such as the one described in Chapters 5 and 6 require far more computation (and memory) early during the localisation process relative to when the user's position has been determined (i.e. during tracking). Although it is unlikely that real-world positioning systems will ever perform global localisation, they may well perform some form of assisted localisation that is still far more expensive than tracking. As a result any mobile device capable of performing such a task would be vastly underutilised during the majority of time during which the user is being tracked.

An interesting alternative to performing all of the computation on a mobile device is to consider offloading computationally expensive operations. For example a mobile device could offload step events to a service provider, whose servers are running a localisation filter on the user's behalf and returning the most likely position following each update. There are various trade-offs associated with such an approach. It may well be desirable to offload computation during the early stages of localisation if the power saved by the mobile device by performing less computation outweighs the additional power required to communicate with the server. If computation is offloaded then it is also the case that the device itself does not need to be able to perform computationally expensive localisation tasks. Although offloading computation during localisation may well save power on a mobile device, the opposite may be true during tracking because far less computation is required to perform each update. In this case the additional power required to communicate with the server may be larger than the power saving that can be achieved by reducing the amount of computation. Hence it may be optimal to migrate a localisation filter from a server to a mobile device at some point during the localisation process.

It is clear that there are many trade-offs involved when dealing with limited electrical power and computational resources. Quantifying these trade-offs is an important area of future research that needs to be investigated before practical systems can be developed based on the algorithms in this thesis.

[1] This figure is the specified rating of the battery in the recently released HTC Nexus 1.

8.2.2.2 User concerns

When deploying practical pedestrian tracking systems it is necessary to consider the concerns of potential users, such as their location privacy and the inconvenience of wearing on-body sensors.

Since the algorithms developed in this thesis use only on-body sensors, it is possible for a user to protect their location privacy by running a localisation filter on their own mobile device. Note however that care must be taken to avoid leaking their location in unintended ways. For example a localisation filter running on a mobile device may leak the user's location if it requests building plan data from a map provider as it is needed.

For many users, wearing on-body sensors is a major inconvenience. One solution might be to build such sensors directly into devices that are already carried by many users. For example foot-mounted sensors could be embedded into the soles of shoes. Inertial sensors are also being embedded within many of today's mobile telephones.

8.2.2.3 Supporting multiple users

The simplest application of the algorithms in this thesis to multiple users is to run a separate localisation filter for each individual. Even when using this approach it may be possible to reduce the cost of tracking each user by sharing resources. In particular consider a localisation and tracking system in which a service provider runs a localisation filter for each user. In normal operation it is likely that at a given time the majority of filters will be tracking users' positions whilst only a few will be performing localisation tasks. Hence the service provider should not need to provision sufficient computational resources to allow every user to execute localisation tasks simultaneously. Further work is needed to establish the extent to which computational resources can be shared and distributed among many users.

In addition to the sharing of computational resources, there are a number of other areas that could be researched further when considering multiple users. For example radio maps for assisted localisation could be built, updated and shared between multiple users. Proximity to other users whose positions are already known could also be used in order to speed up the localisation process in systems supporting multiple users.

8.3 Final words

MEMS inertial sensors can be used as an alternative means of tracking people in indoor environments. It is possible to both determine and subsequently track the absolute position of

a pedestrian by combining relative movement information obtained from such sensors with knowledge of environmental constraints provided in the form of a map. Such an approach does not rely on any fixed infrastructure and hence can be used in situations where it is impractical or impossible to install such infrastructure into the environment in advance. Furthermore this approach has been shown to track users to sub-metre accuracies in a typical office environment, which is sufficient for many location-aware applications.

The feasibility of applying the techniques presented in this thesis will increase as the quality of MEMS inertial sensors continues to improve, detailed maps of indoor environments become more commonly available and the computational power of mobile devices continues to increase. Although it is likely that some form of constraints or external measurements will always be required to track pedestrians using inertial sensors, it may be possible to use more general constraints as MEMS inertial sensors continue to improve. In particular it may be possible to relax the requirement for a sensor to be worn on the user's foot.

Appendix A

Analysing noise in MEMS inertial sensors

Section 3.1 describes a number of noise processes that perturb signals from micro-machined electromechanical systems (MEMS) inertial sensors. This appendix describes a technique known as Allan Variance that can be used to detect and quantify such processes. Allan Variance is a time domain analysis technique originally designed for characterising noise and stability in clock systems. The technique can be applied to any time-series (a sequence of data-points measured at times separated by uniform intervals) to determine the characteristics of its underlying noise processes. The Allan Variance of a time-series is a function of averaging time t. For a given t, the Allan Variance is calculated as follows:

1. Take a long sequence of data and divide it into bins of length t. There must be enough data for at least nine bins, otherwise the results obtained begin to lose their significance [98].

2. Average the data in each bin to obtain a list of averages $(a(t)_1, a(t)_2, ..., a(t)_n)$, where n is the number of bins.

3. The Allan Variance is then given by

$$\mathrm{AVAR}(t) = \frac{1}{2 \cdot (n-1)} \sum_{i=1}^{n-1} \left(a(t)_{i+1} - a(t)_i\right)^2 . \tag{A.1}$$

To determine the characteristics of the underlying noise processes, the Allan Deviation

$$\mathrm{AD}(t) = \sqrt{\mathrm{AVAR}(t)} \tag{A.2}$$

is plotted as a function of t on a log-log scale. Different types of random process cause slopes with different gradients to appear on the plot, as shown in Figure A.1. Furthermore different processes usually appear in different regions of t, allowing their presence to be easily

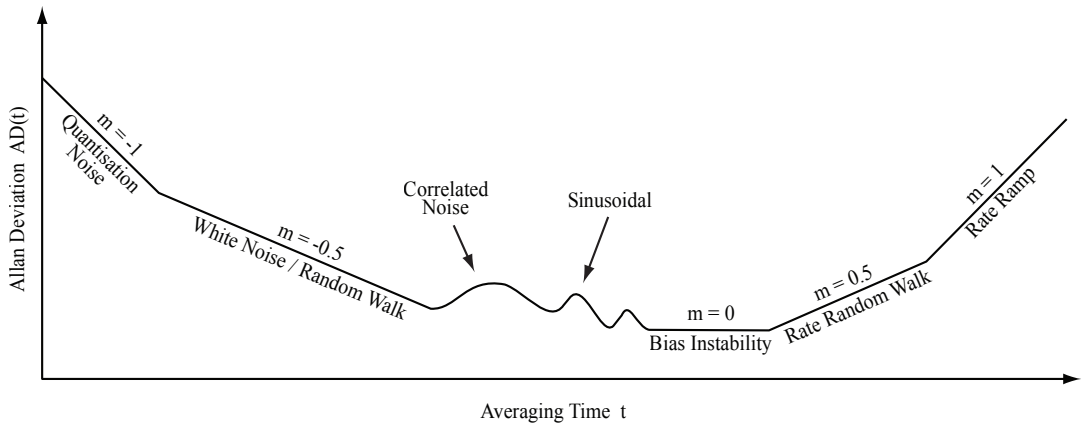


Figure A.1: A possible log-log plot of Allan Deviation [99].

identified. Having identified a process it is then possible to read its numerical parameters directly from the plot. For a MEMS inertial measurement unit (IMU) such as the Xsens Mtx the most significant noise processes are white noise processes that cause angle-random-walk (ARW) and velocity-random-walk (VRW) errors in integrated gyroscope and accelerometer signals respectively, and bias instability processes that cause the signal biases to wander over time. These processes can be identified and read from the Allan Deviation plot as follows:

- A white noise process appears on the Allan Deviation plot as a slope with gradient -0.5. The corresponding root power spectral density (PSD) n is read directly from the graph by fitting a straight line through the slope and reading its value at $t = 1$.
- A bias instability process appears on the plot as a flat region around the minimum. Its numerical value is the minimum value on the Allan Deviation curve.

A full description of the Allan Variance technique and how it can be used to identify and quantify other random processes can be found in [99].

A.1 Xsens Mtx analysis

Figure A.2 shows the Allan Deviation curves for the x, y and z axis gyroscopes of an Xsens Mtx. The plot was generated from a 12 hour log of measurement data gathered from the IMU whilst it was kept stationary. The sampling frequency of the device was set to $100\,\mathrm{Hz}$. The slopes on the left side of the plot have gradients that are almost exactly equal to -0.5, indicating the presence of white noise as expected. The numerical values read from the plot for both white noise and bias instability processes are listed in Table A.1.

Figure A.3 shows the Allan Deviation curves for the three accelerometers in the Mtx. The z-axis accelerometer is perturbed by a white noise process that is almost twice that of the other two accelerometers. Initially it was thought that this may be due to a gravitational

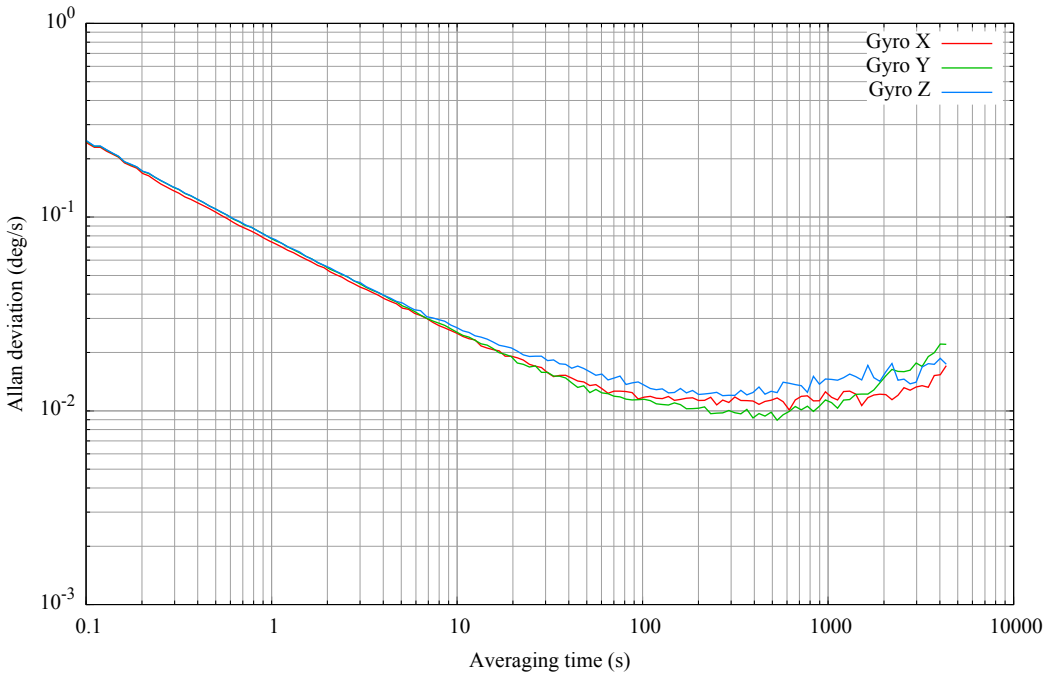


Figure A.2: Allan Deviation curves for the gyroscopes of an Xsens Mtx.

	Bias instability	Random noise n (root PSD)
X Axis	$0.010°/\text{s}$ (at 620 s)	$0.075°/\text{s}/\sqrt{\text{Hz}}$
Y Axis	$0.009°/\text{s}$ (at 530 s)	$0.078°/\text{s}/\sqrt{\text{Hz}}$
Z Axis	$0.012°/\text{s}$ (at 270 s)	$0.079°/\text{s}/\sqrt{\text{Hz}}$

Table A.1: Gyroscope noise measurements for the Xsens Mtx.

effect (since the log data was obtained with the sensor's z-axis pointing upwards in the global frame-of-reference), however the same curve was obtained when the analysis was repeated on data obtained with the z-axis aligned horizontally. Hence it was concluded that the z-axis accelerometer was for some reason noisier than the other accelerometers in the tested device. Table A.2 lists the numerical values read from the Allan Deviation plot.

A.2 Conversion to sigma values

When modelling or simulating angle or velocity random walks and bias instability effects it is usually necessary to convert noise measurements into the 1σ values of their underlying noise processes. The most common situation where this is necessary is when modelling such processes in a Kalman filter, but it is also necessary when simulating noise in an INS simulator [79].

The standard deviation of the random noise perturbing each measurement is obtained from

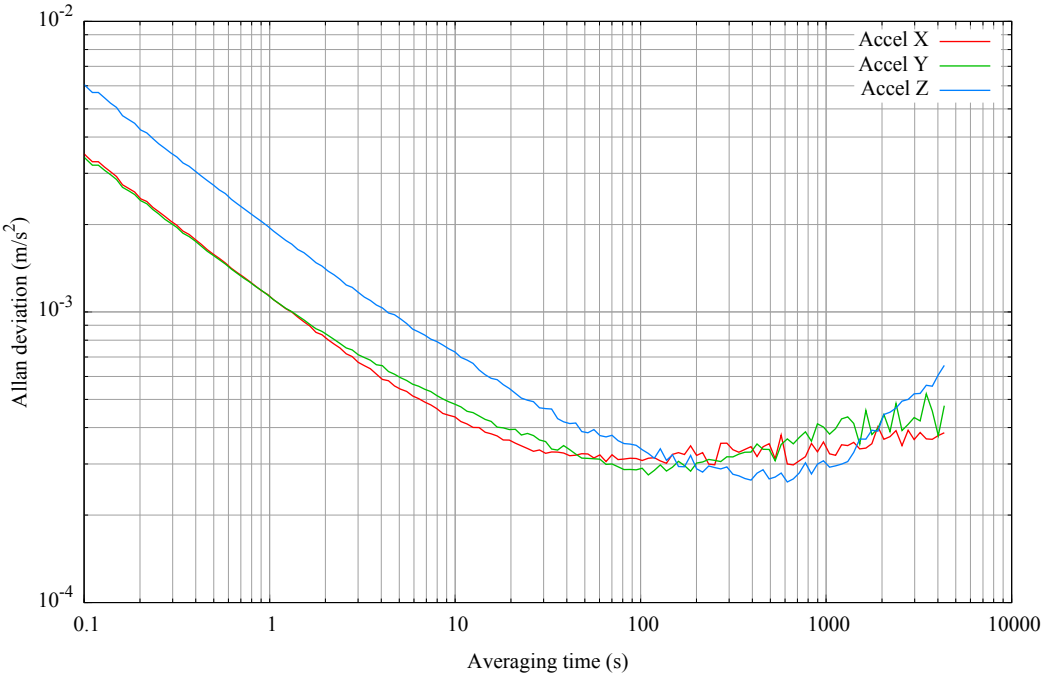


Figure A.3: Allan Deviation curves for the accelerometers of an Xsens Mtx.

	Bias instability	Random noise n (root PSD)
X Axis	$3.0 \cdot 10^{-4}$ m/s^2 (at 670 s)	0.0011 m/s^2/$\sqrt{\text{Hz}}$
Y Axis	$2.8 \cdot 10^{-4}$ m/s^2 (at 110 s)	0.0011 m/s^2/$\sqrt{\text{Hz}}$
Z Axis	$2.6 \cdot 10^{-4}$ m/s^2 (at 620 s)	0.0020 m/s^2/$\sqrt{\text{Hz}}$

Table A.2: Accelerometer noise measurements for the Xsens Mtx.

the root PSD n by dividing by the root of the sampling period

$$\sigma = \frac{n}{\sqrt{\delta t}} \tag{A.3}$$

where δt is the sampling period of the device ($\delta t = 0.01\,\text{s}$ for the Xsens Mtx tested in this appendix). See Section 3.1.1.2 for more detail.

A bias instability of magnitude BS over a duration t is usually modelled as a random walk perturbing the bias of the sensor. The standard deviation of each random variable in the underlying white noise sequence is given by

$$\sigma = \frac{\text{BS}}{\sqrt{\delta t \cdot t}}\,. \tag{A.4}$$

The converted 1σ values for the Xsens Mtx are shown in Table A.3.

	Bias instability (σ value of the underlying white noise sequence)	Random noise (σ value)
X-Gyro ($^\circ/\mathrm{s}$)	4.0×10^{-3}	0.75
Y-Gyro ($^\circ/\mathrm{s}$)	3.9×10^{-3}	0.78
Z-Gyro ($^\circ/\mathrm{s}$)	7.3×10^{-3}	0.79
X-Accel ($\mathrm{m/s^2}$)	1.2×10^{-4}	0.011
Y-Accel ($\mathrm{m/s^2}$)	2.7×10^{-4}	0.011
Z-Accel ($\mathrm{m/s^2}$)	1.0×10^{-4}	0.020

Table A.3: The 1σ values of white noise sequences that underlie bias instability and random noise processes for an Xsens Mtx.

Appendix B

Rotation matrices

A rotation matrix is any orthogonal matrix whose determinant is equal to 1. The effect of multiplying a vector by a rotation matrix is to change its direction but not its magnitude. In a right-handed 3-dimensional Cartesian co-ordinate system, rotations about the x, y and z axes are achieved using the rotation matrices

$$\boldsymbol{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \tag{B.1}$$

$$\boldsymbol{R}_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \tag{B.2}$$

$$\boldsymbol{R}_z = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{B.3}$$

which are defined such that a positive rotation about an axis occurs in the clockwise direction when looking along that axis from the origin [13].

Any rotation $\boldsymbol{R}$ can be expressed in the form

$$\boldsymbol{R} = \boldsymbol{R}_x \boldsymbol{R}_y \boldsymbol{R}_z \tag{B.4}$$

with suitable values of ϕ, θ and ψ. It is however important to note that the order in which the rotations are applied is important, since in general rotations in a 3-dimensional space are non-commutative.

B.1 Small angle approximation

If ϕ, θ and ψ are small, then $\sin\phi \rightarrow \phi$, $\sin\theta \rightarrow \theta$, $\sin\psi \rightarrow \psi$ and the cosines of ϕ, θ and ψ tend to 1. Using these approximations and ignoring the products of angles (since they are small), the order in which rotations around the individual axes are applied becomes unimportant and the rotation matrix $\boldsymbol{R}$ becomes

$$\boldsymbol{R} = \boldsymbol{R}_x\boldsymbol{R}_y\boldsymbol{R}_z \approx \begin{pmatrix} 1 & \psi & -\theta \\ -\psi & 1 & \phi \\ \theta & -\phi & 1 \end{pmatrix}. \tag{B.5}$$

This is known as the small angle approximation.

B.2 Rotation about an arbitrary axis

The rotation matrix that defines a clockwise rotation (when looking along the axis from the origin) of magnitude θ about an arbitrary unit-vector $\hat{\boldsymbol{r}} = (r_x, r_y, r_z)$ that passes through the origin is given by

$$\boldsymbol{R} = \begin{pmatrix} tr_x^2 + c & tr_xr_y + sr_z & tr_xr_z - sr_y \\ tr_xr_y - sr_z & tr_y^2 + c & tr_yr_z + sr_x \\ tr_xr_z + sr_y & tr_yr_z - sr_x & tr_z^2 + c \end{pmatrix} \tag{B.6}$$

where $s = \sin\theta$, $c = \cos\theta$ and $t = 1 - \cos\theta$ [100]. If θ is small then small angle approximations can be used to give

$$\boldsymbol{R} \approx \begin{pmatrix} 1 & r_z\theta & -r_y\theta \\ -r_z\theta & 1 & r_x\theta \\ r_y\theta & -r_x\theta & 1 \end{pmatrix}. \tag{B.7}$$

B.3 Aligning arbitrary vectors

It is sometimes necessary to calculate a rotation matrix $\boldsymbol{R}$ that will rotate some vector $\boldsymbol{v}_1$ so that it becomes aligned with another vector $\boldsymbol{v}_2$. Such a matrix can be derived as follows:

1. Calculate the normalised vectors $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$.
2. Calculate the cross-product $\boldsymbol{r} = \hat{\boldsymbol{v}}_1 \times \hat{\boldsymbol{v}}_2$. The resulting vector is perpendicular to both of the original vectors and has a magnitude

$$|\boldsymbol{r}| = \sin\theta \tag{B.8}$$

where θ is the angle between them.

3. A suitable value of $\boldsymbol{R}$ is a rotation matrix that corresponds to a clockwise rotation of magnitude θ about $\hat{\boldsymbol{r}}$, the exact form of which is given in Appendix B.2.

Note that additional care must be taken to detect and handle the special cases in which $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are parallel (in which case $\boldsymbol{R} = \boldsymbol{I}$) and in which $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are anti-parallel (in which case a suitable value of $\boldsymbol{R}$ is any rotation matrix that corresponds to a rotation of magnitude π about an axis perpendicular to $\boldsymbol{v}_1$).

Appendix C

Kalman filters

Kalman filters represent the Bayesian belief $\mathrm{Bel}(\boldsymbol{x}_t)$ as a Gaussian distribution defined by its mean[1] $\hat{\boldsymbol{x}}_t$ and covariance $\boldsymbol{P}_t$. They assume that the true state $\boldsymbol{x}$ of the system evolves according to the linear equation

$$\boldsymbol{x}_t = \boldsymbol{F}_t \boldsymbol{x}_{t-\delta t} + \boldsymbol{w}_t \tag{C.1}$$

where $\boldsymbol{w}_t$ is process noise which is assumed to be zero mean Gaussian distributed with known covariance $\boldsymbol{Q}_t$:

$$\boldsymbol{w}_t = \mathcal{N}_{0,\boldsymbol{Q}_t}\,. \tag{C.2}$$

Measurements are assumed to occur at discrete points in time according to the linear relationship

$$\boldsymbol{z}_t = \boldsymbol{H}_t \boldsymbol{x}_t + \boldsymbol{v}_t \tag{C.3}$$

where $\boldsymbol{v}_t$ is measurement noise, also assumed to be zero mean Gaussian distributed

$$\boldsymbol{v}_t = \mathcal{N}_{0,\boldsymbol{R}_t}\,. \tag{C.4}$$

Given these assumptions, the optimal Bayesian filter propagation and correction equations can be implemented using computationally efficient matrix operations [53, 52]. The estimated state is represented by its mean $\hat{\boldsymbol{x}}_t$ and covariance $\boldsymbol{P}_t$ and is propagated using the equations

$$\hat{\boldsymbol{x}}_t^- = \boldsymbol{F}_t \hat{\boldsymbol{x}}_{t-\delta t} \tag{C.5}$$

$$\boldsymbol{P}_t^- = \boldsymbol{F}_t \boldsymbol{P}_{t-\delta t} \boldsymbol{F}_t^T + \boldsymbol{Q}_t\,. \tag{C.6}$$

[1] A hat symbol is used to distinguish the estimated mean state $\hat{\boldsymbol{x}}_t$ from the true state of the system $\boldsymbol{x}_t$.

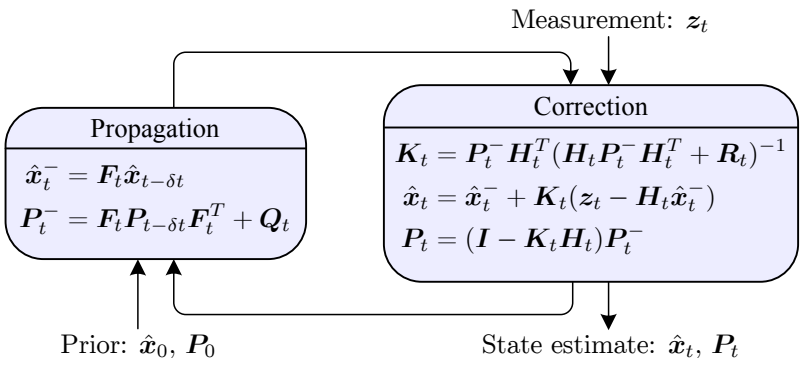


Figure C.1: The Kalman filter.

A measurement $\boldsymbol{z}_t$ is used to correct the propagated state using the equations

$$\boldsymbol{K}_t = \boldsymbol{P}_t^-\boldsymbol{H}_t^T(\boldsymbol{H}_t\boldsymbol{P}_t^-\boldsymbol{H}_t^T + \boldsymbol{R}_t)^{-1} \tag{C.7}$$

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t^- + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{H}_t\hat{\boldsymbol{x}}_t^-) \tag{C.8}$$

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{P}_t^- \,. \tag{C.9}$$

Multiple measurements can be applied at a single point in time by repeating the correction step with $\hat{\boldsymbol{x}}_t^- = \hat{\boldsymbol{x}}_t$. The complete Kalman filter is shown in Figure C.1.

C.1 Linearised Kalman filters

The standard Kalman filter assumes that both the system being estimated and any measurements can be described by linear equations of the form

$$\boldsymbol{x}_t = \boldsymbol{F}_t\boldsymbol{x}_{t-\delta t} + \boldsymbol{w}_t \tag{C.10}$$

$$\boldsymbol{z}_t = \boldsymbol{H}_t\boldsymbol{x}_t + \boldsymbol{v}_t \,. \tag{C.11}$$

A linearised Kalman filter relaxes these assumptions by linearising the more general equations

$$\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_{t-\delta t}) + \boldsymbol{w}_t \tag{C.12}$$

$$\boldsymbol{z}_t = \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t) + \boldsymbol{v}_t \,. \tag{C.13}$$

about some approximate trajectory $\boldsymbol{x}_t^*$, which is assumed to be known [53]. The functions $\boldsymbol{\mathcal{F}}_t$ and $\boldsymbol{\mathcal{H}}_t$ are known state transition and measurement equations, which must be differentiable but may be non-linear.

The true state of the system can be expressed in terms of the approximate trajectory and its error

$$\boldsymbol{x}_t = \boldsymbol{x}_t^* + \delta\boldsymbol{x}_t \,. \tag{C.14}$$

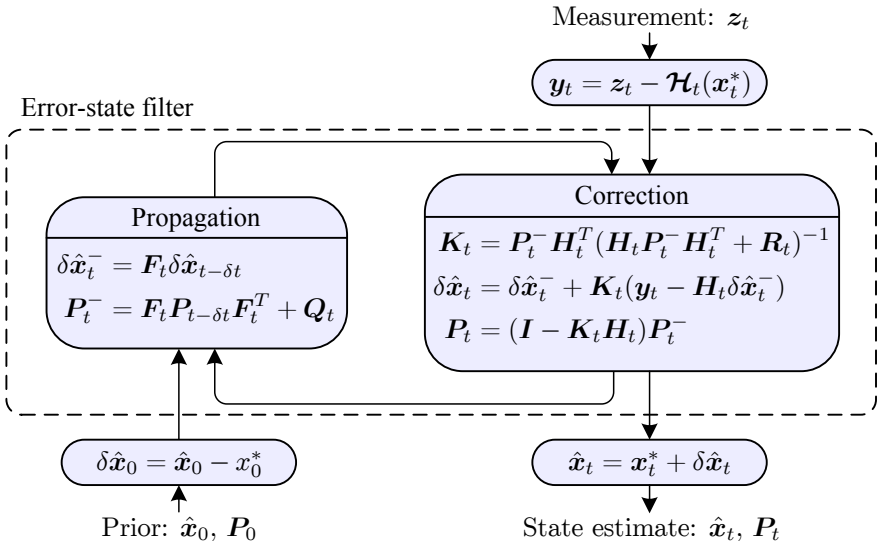


Figure C.2: The linearised Kalman filter.

Hence Equations C.12 and C.13 can be re-written as

$$\boldsymbol{x}_t^* + \delta\boldsymbol{x}_t = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_{t-\delta t}^* + \delta\boldsymbol{x}_{t-\delta t}) + \boldsymbol{w}_t \tag{C.15}$$

$$\boldsymbol{z}_t = \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t^* + \delta\boldsymbol{x}_t) + \boldsymbol{v}_t\,. \tag{C.16}$$

Assuming $\delta\boldsymbol{x}_t$ is small, the functions $\boldsymbol{\mathcal{F}}_t$ and $\boldsymbol{\mathcal{H}}_t$ can approximated by the first-order terms of their Taylor's series expansions, giving

$$\boldsymbol{x}_t^* + \delta\boldsymbol{x}_t \approx \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_{t-\delta t}^*) + \left[\frac{\partial\boldsymbol{\mathcal{F}}_t}{\partial\boldsymbol{x}}\right]_{\boldsymbol{x}_{t-\delta t}^*} \cdot \delta\boldsymbol{x}_{t-\delta t} + \boldsymbol{w}_t \tag{C.17}$$

$$\boldsymbol{z}_t \approx \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t^*) + \left[\frac{\partial\boldsymbol{\mathcal{H}}_t}{\partial\boldsymbol{x}}\right]_{\boldsymbol{x}_t^*} \cdot \delta\boldsymbol{x}_t + \boldsymbol{v}_t\,. \tag{C.18}$$

Assuming that for the approximate trajectory $\boldsymbol{x}_t^* = \boldsymbol{\mathcal{F}}_t(\boldsymbol{x}_{t-\delta t}^*)$, we can re-arrange Eqs. C.17 and C.18 to obtain the linear equations

$$\delta\boldsymbol{x}_t = \left[\frac{\partial\boldsymbol{\mathcal{F}}_t}{\partial\boldsymbol{x}}\right]_{\boldsymbol{x}_{t-\delta t}^*} \cdot \delta\boldsymbol{x}_{t-\delta t} + \boldsymbol{w}_t \tag{C.19}$$

$$\boldsymbol{z}_t - \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t^*) = \left[\frac{\partial\boldsymbol{\mathcal{H}}_t}{\partial\boldsymbol{x}}\right]_{\boldsymbol{x}_t^*} \cdot \delta\boldsymbol{x}_t + \boldsymbol{v}_t \tag{C.20}$$

which are in the form required by the standard Kalman filter (Eqs. C.1 and C.3). Hence a linearised Kalman filter is implemented as a standard Kalman filter that maintains a belief $\mathrm{Bel}(\delta\boldsymbol{x}_t)$ of the error-state. As in the standard Kalman filter the estimated state is represented

by its mean $\delta\hat{\boldsymbol{x}}_t$ and covariance $\boldsymbol{P}_t$. The error-state transition matrix is given by

$$\boldsymbol{F}_t = \left[\frac{\partial \boldsymbol{\mathcal{F}}_t}{\partial \boldsymbol{x}}\right]_{\boldsymbol{x}^*_{t-\delta t}} . \tag{C.21}$$

By comparing Equations C.20 and C.3 it is clear that a measurement $\boldsymbol{z}_t$ of the system's state can be incorporated by the filter in the form of an error-state measurement

$$\boldsymbol{y}_t = \boldsymbol{z}_t - \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}^*_t) \tag{C.22}$$

with measurement matrix

$$\boldsymbol{H}_t = \left[\frac{\partial \boldsymbol{\mathcal{H}}_t}{\partial \boldsymbol{x}}\right]_{\boldsymbol{x}^*_t} . \tag{C.23}$$

The complete linearised Kalman filter is shown in Figure C.2.

C.2 Extended Kalman filters

The linearised Kalman filter is suitable for non-linear systems where an approximate trajectory of the state can be precomputed. For example when a missile is to be tracked as it moves towards a static target, its planned trajectory is known prior to its launch. The extended Kalman filter is suitable for non-linear systems in which an approximate trajectory is not known in advance. It is identical to the linearlised Kalman filter, except that the linearisation takes place about the filter's estimated trajectory $\hat{\boldsymbol{x}}_t$, rather than the precomputed trajectory $\boldsymbol{x}^*_t$ [53]. The estimated trajectory is propagated using

$$\hat{\boldsymbol{x}}^-_t = \boldsymbol{\mathcal{F}}_t(\hat{\boldsymbol{x}}_{t-\delta t}) . \tag{C.24}$$

After each correction step the estimated error-state $\delta\hat{\boldsymbol{x}}_t$ is used to correct the trajectory

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}^-_t + \delta\hat{\boldsymbol{x}}_t \tag{C.25}$$

before being reset to zero. Hence $\delta\hat{\boldsymbol{x}}_t$ is equal to zero when entering both the propagation and correction steps, allowing the corresponding equations to be simplified as shown in Figure C.3. Since the filter is linearised about the estimated trajectory, $\boldsymbol{F}_t$ and $\boldsymbol{H}_t$ are the Jacobian matrices

$$\boldsymbol{F}_t = \left[\frac{\partial \boldsymbol{\mathcal{F}}_t}{\partial \boldsymbol{x}}\right]_{\hat{\boldsymbol{x}}_{t-\delta t}} \tag{C.26}$$

$$\boldsymbol{H}_t = \left[\frac{\partial \boldsymbol{\mathcal{H}}_t}{\partial \boldsymbol{x}}\right]_{\hat{\boldsymbol{x}}^-_t} . \tag{C.27}$$

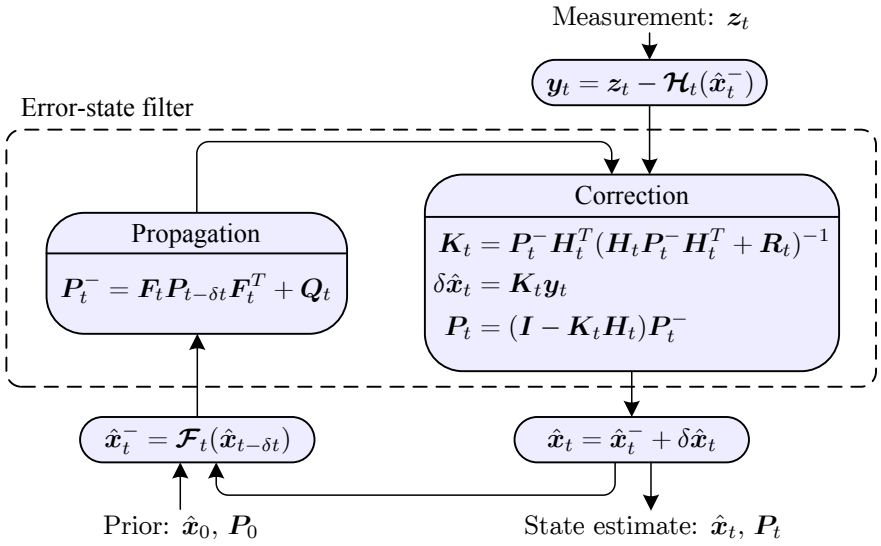


Figure C.3: The extended Kalman filter.

C.2.1 Alternative formulation

The extended Kalman filter as described above consists of a regular Kalman filter operating on the error-state whilst the state of the system is propagated separately. It is often more convenient to track the estimated state of the system directly within the filter. This can be achieved using an alternative formulation of the extended Kalman filter in which the propagation equations are

$$\hat{\boldsymbol{x}}_t^- = \boldsymbol{\mathcal{F}}_t(\hat{\boldsymbol{x}}_{t-\delta t}) \tag{C.28}$$

$$\boldsymbol{P}_t^- = \boldsymbol{F}_t \boldsymbol{P}_{t-\delta t} \boldsymbol{F}_t^T + \boldsymbol{Q}_t \tag{C.29}$$

and the measurement update equations are

$$\boldsymbol{K}_t = \boldsymbol{P}_t^- \boldsymbol{H}_t^T (\boldsymbol{H}_t \boldsymbol{P}_t^- \boldsymbol{H}_t^T + \boldsymbol{R}_t)^{-1} \tag{C.30}$$

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t^- + \boldsymbol{K}_t (\boldsymbol{z}_t - \boldsymbol{\mathcal{H}}_t(\hat{\boldsymbol{x}}_t^-)) \tag{C.31}$$

$$\boldsymbol{P}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t) \boldsymbol{P}_t^- \,. \tag{C.32}$$

Note that these equations can be obtained simply by incorporating the three external operations shown in Figure C.3 directly into the propagation and correction steps. The resulting filter is shown in Figure C.4.

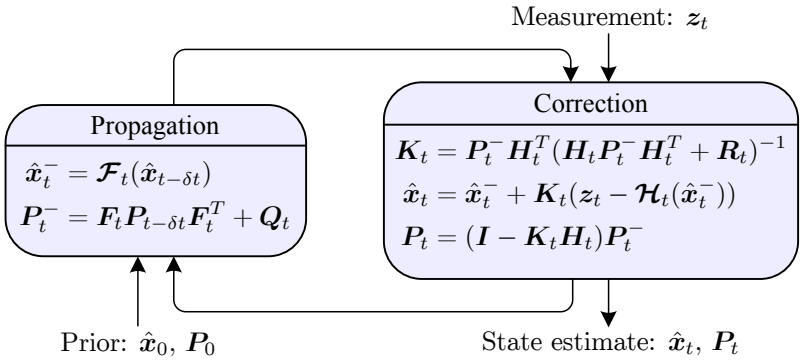


Figure C.4: The extended Kalman filter (alternative formulation).

Appendix D

Ultrasonic correction of inertial navigation systems

This appendix describes how range measurements between an inertial measurement unit (IMU) and known locations in an environment can be used to correct an inertial navigation system (INS) that is tracking the IMU's state (i.e. its position, velocity and orientation). Such range measurements can be obtained by timing ultrasonic pulses, which can either be emitted by beacons in the environment and detected by a receiver attached to the IMU (unilateral positioning) [8, 29], or emitted from the IMU and detected by receivers in the environment (multilateral positioning) [28, 6]. For more details about ultrasonic positioning systems see Section 2.1.2.2. Note that the approach presented here is broadly applicable to range measurements obtained using other positioning technologies, with the caveat that the error characteristics of such technologies may differ from those of ultrasound.

D.1 Applying range measurements

Section 3.4 describes a general purpose error-state Kalman filter for probabilistically correcting the state of an INS. It is able to use any measurement $\boldsymbol{z}_t$ to correct an INS, provided that it can be related to the state $\boldsymbol{x}_t$ of the INS by an equation of the form

$$\boldsymbol{z}_t = \boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t) + \boldsymbol{v}_t \,. \tag{D.1}$$

where $\boldsymbol{\mathcal{H}}_t$ is a differentiable (but possibly non-linear) function and $\boldsymbol{v}_t = \mathcal{N}_{0,\boldsymbol{R}_t}$ is zero mean Gaussian distributed noise with a known covariance $\boldsymbol{R}_t$. Hence to correct an INS using ultrasonic range measurements it is sufficient to implement this filter and specify a function $\boldsymbol{\mathcal{H}}_t$ and a covariance matrix $\boldsymbol{R}_t$ that are suitable for such measurements. This is the approach

adopted below. Note that it is assumed that the reader of this appendix is familiar with both the Kalman filter described in Section 3.4 and with the notation introduced in Section 3.2.

When a microphone (or transducer for multilateral positioning) is attached to an IMU it will usually be offset from the IMU's origin (i.e. the point of the IMU that is tracked by an INS). The offset between the microphone and the origin is constant in the body frame (i.e. the IMU's local frame of reference) and is denoted $\boldsymbol{o}^b$. Given that $\boldsymbol{o}^b$ is known, consider a range measurement $\boldsymbol{z}_t = d_t$ between the IMU and a beacon (or receiver for multilateral positioning) at a known location $\boldsymbol{r} = (r^{g_x}, r^{g_y}, r^{g_z})$ in the environment. The measurement can be related to the state of the INS by the function

$$\boldsymbol{\mathcal{H}}_t(\boldsymbol{x}_t) = \sqrt{(r^{g_x} - i_t)^2 + (r^{g_y} - j_t)^2 + (r^{g_z} - k_t)^2} \tag{D.2}$$

where

$$(i_t, j_t, k_t) = \boldsymbol{s}_t + \boldsymbol{C}_t \boldsymbol{o}^b \tag{D.3}$$

is the position of the microphone (or transducer) in the local navigation frame, in which $\boldsymbol{C}_t$ and $\boldsymbol{s}_t$ are the orientation and displacement of the IMU respectively. The Jacobian matrix of partial derivatives that is required by the filter is given by

$$\boldsymbol{H}_t = \left[\frac{\partial \boldsymbol{\mathcal{H}}}{\partial \boldsymbol{x}}\right]_{\hat{\boldsymbol{x}}_t^-} = (0, 0, 0, 0, 0, 0, \frac{(\hat{i}_t^- - r^{g_x})}{\mathcal{H}_t(\hat{\boldsymbol{x}}_t^-)}, \frac{(\hat{j}_t^- - r^{g_x})}{\mathcal{H}_t(\hat{\boldsymbol{x}}_t^-)}, \frac{(\hat{k}_t^- - r^{g_z})}{\mathcal{H}_t(\hat{\boldsymbol{x}}_t^-)}) \tag{D.4}$$

in which

$$(\hat{i}_t^-, \hat{j}_t^-, \hat{k}_t^-)^T = \hat{\boldsymbol{s}}_t^- + \hat{\boldsymbol{C}}_t^- \boldsymbol{o}\,. \tag{D.5}$$

is the position of the microphone (or transducer) as estimated by the current state of the INS.

All that remains is to specify a suitable covariance matrix $\boldsymbol{R}_t$ that describes uncertainty in the measurement. For a range measurement this matrix is a 1×1 matrix containing a single variance, which should be set to a value appropriate for the ultrasonic system that is being used (the accuracy of such systems is dependent on many factors, for example whether they take air temperature into account when converting a time-of-flight into a corresponding range measurement).

D.2 Filtering range measurements

One problem that has not been addressed above is that of signal reflections, which cause range measurements that are larger than the true distance between the IMU and the beacon (or receiver in multilateral positioning) in the environment. Incorporating such measurements into the INS solution will degrade its accuracy. To avoid this range measurements that correspond

to reflected signals should be filtered out whenever possible. There are two approaches to filtering out such measurements, which should be used together to obtain best performance:

- If multiple signals are detected by a receiver for a single transmitted pulse, then only the first can correspond to a direct signal. Hence the time-of-flight corresponding to the first signal should be converted into a range measurement, whilst all of the secondary signals should be ignored.
- Before a range measurement is applied it can be compared to the range that is predicted by the current INS state estimate. If range measurements are applied frequently then this estimate should be a good estimate of the true range (since the estimated position of the IMU should be a good estimate of its true position). Hence if the measurement is much larger than the predicted range then it is likely that it corresponds to a reflected signal and hence should be discarded. It is also beneficial to discard range measurements that are much shorter than expected. Although they cannot be caused by signal reflections, they can be caused by ultrasonic noise in the environment. The thresholds over which ranges are discarded should be chosen based on the accuracy of the ultrasonic system that is used and can also take the uncertainty in the current state estimate into account (e.g. by allowing the measured range to differ from the predicted range by a larger amount when there is more uncertainty in the estimated position of the IMU).

D.3 Unilateral versus multilateral positioning

When designing a combined inertial and ultrasonic tracking system from scratch it is always preferable to design the ultrasonic component to be unilateral, meaning that beacons in the environment transmit ultrasonic pulses that are detected by a microphone that is attached to the IMU. The main benefit compared to a multilateral design is that all of the required measurements end up at the same place (i.e. at the IMU). Hence if the software components of the system are executed locally on the mobile device, then there is no need to transmit any measurements from one place to another. In contrast in multilateral positioning range measurements are calculated by receivers in the environment. Hence it is necessary to transmit them back to the tracked object, or offload IMU measurements if the software is run elsewhere. A secondary benefit of a unilateral design is that the range obtained from a time-of-flight is a measurement of the distance between the IMU and a beacon at the *precise* moment that the pulse is received. In multilateral positioning a range calculated by a receiver is a measurement of this distance at the point in time when the pulse was transmitted from the IMU, which is always in the *past*. Since Bayesian filters are only able to incorporate measurements at the current time, the filter must run with a small delay (equal to the largest time-of-flight that might be measured) if a multilateral design is used. This additional delay is not needed in the

unilateral case and is significant if low latency pose estimates are required (e.g. for use by virtual reality applications).

D.4 PDR correction using the Active Bat system

This section describes the extension of the pedestrian dead reckoning (PDR) filter described in Section 4.1 to incorporate ultrasonic range measurements obtained from the Active Bat positioning system. The extended filter provides a drift free estimate of the position of a tracked user. Heading estimates calculated by the extended filter are also drift free. This is because the error-state Kalman filter is able to correct correlated errors in different components of the INS state, as described in Section 3.4. Heading errors are strongly correlated to errors in position (an error in heading will always result in an error in position unless the IMU remains stationary) and hence range measurements prevent drift from accumulating in the estimated heading as well as in the estimated position. The data obtained from the extended filter was used as ground truth in the analysis of the (unextended) PDR filter, as described in Section 4.2.2.

The Active Bat system is a multilateral ultrasonic location system that was initially developed at AT&T laboratories in Cambridge [28, 6]. The system is currently deployed in one wing of the William Gates building. Ideally a unilateral system would have been used to provide ground truth measurements, however in this case it was necessary to use what was already installed in the building. Since the ground truth data obtained was required only for testing purposes, it was sufficient to log all data and replay it to the filter later. Hence the latency issue described in Section D.3 was avoided. The problem of getting all of the measurements to the same place was also avoided by logging IMU measurements on an ultra mobile PC (UMPC) that was carried by the user and logging Active Bat measurements on a desktop PC that was connected to the Active Bat system. Both logs were then transferred to the same machine for processing.

The PDR filter was extended by modifying its error-state Kalman filter to support the application of range measurements, as described in Section D.1. The Active Bat system was set to query the position of a single bat 25 times per second, which was attached to the IMU worn on the user's foot directly above the IMU (see Figure 4.8 on page 76). Range measurements obtained from the Active Bat system were used to correct the INS, provided that they passed the time synchronisation and filtering conditions outlined below. All range measurements that were applied were assumed to have equal uncertainty of $\sigma = 0.02\,\mathrm{m}$, which was empirically determined to give good performance (and is roughly equal to the expected accuracy of the system).

D.4.1 Time synchronisation

One major problem encountered when applying Active Bat range measurements was synchronising the mobile device that logged the IMU measurements with the desktop PC that logged measurements from the Active Bat system. Such synchronisation was necessary to ensure that measurements from the different systems were replayed into the filter at the correct points in time. This was achieved using Chrony[1], which synchronises machines using the Network Time Protocol (NTP) and additionally constructs locally stored models for each clock that are used to minimise clock drift when the machines cannot contact one another. The two logging machines were synchronized over a wired LAN before each data collection run.

During each run various factors can cause synchronisation issues to arise. Firstly, the two clocks can be expected to drift apart slightly despite the corrections applied by Chrony. To minimise this effect each data collection run was limited to at most 15 minutes, during which the drift between the two clocks would be in the order of milliseconds. Secondly, small delays between each measurement being made and subsequently being timestamped by the logging application can have an effect. Active Bat data must also propagate over a wired LAN before reaching the logging machine. To mitigate these issues range measurements obtained from the Active Bat system were only used if they occurred during a stance phase, as described in Section 4.1.1. Since the user's foot (and hence the attached IMU and Active Bat) is stationary during such periods, small synchronisation errors have no effect. The way in which stance phases are detected by the PDR filter means that there is a window of $100\,\mathrm{ms}$ either side of a stance phase during which the user's foot is stationary (see Figure 4.1 on page 68). Hence this approach allows for drift between the timestamps recorded in the two logs of up to $100\,\mathrm{ms}$, which is an order of magnitude greater than the maximum drift that could be expected.

D.4.2 Filtering range measurements

Secondary signals (i.e. those that arrive after another signal corresponding to the same transmitted pulse) are automatically discarded by the Active Bat system. Range measurements were also filtered based on their consistency with the filter's current state estimate. For a given range measurement the predicted range is calculated as the distance between the receiver and the predicted bat position given by Equation D.5. It was found that rejecting any range measurement that differed from the predicted range by more than $0.06\,\mathrm{m}$ gave good performance. Note that in a system intended for frequent use a more advanced approach may be required to ensure good performance in all cases, however for the limited use required here the simple approach described was found to be sufficient.

[1] http://chrony.tuxfamily.org

Appendix E

Error ellipses

An error ellipse is a simple way of graphically representing the uncertainty of the position of a 2-dimensional point [101]. It is assumed that the uncertainty is Gaussian distributed and hence can be defined by its mean $\boldsymbol{\mu} = (\mu_x, \mu_y)$ and covariance

$$\begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}. \tag{E.1}$$

An error ellipse encodes the maximum and minimum standard deviations of such a distribution as well as their associated directions. These parameters can be calculated by considering the value of the covariance matrix in a rotated coordinate frame (m, n). In this coordinate frame the covariance matrix is given by

$$\begin{bmatrix} \sigma_m^2 & \sigma_{mn} \\ \sigma_{mn} & \sigma_n^2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{E.2}$$

where θ is the angle between the original coordinate frame (x, y) and the rotated frame (m, n) in the clockwise direction. When multiplied out Equation E.2 gives

$$\sigma_m^2 = \sigma_x^2 \cos^2\theta - 2\sigma_{xy} \sin\theta \cos\theta + \sigma_y^2 \sin^2\theta \tag{E.3}$$

$$\sigma_n^2 = \sigma_x^2 \sin^2\theta + 2\sigma_{xy} \sin\theta \cos\theta + \sigma_y^2 \cos^2\theta\,. \tag{E.4}$$

To find the directions of the maximum and minimum standard deviations it is necessary to find the values of θ at the turning points of σ_m^2 or σ_n^2. Arbitrarily choosing σ_n^2 gives

$$\frac{d}{d\theta}\sigma_n^2 = 0 = 2\sigma_x^2 \sin\theta \cos\theta - 2\sigma_y^2 \sin\theta \cos\theta + 2\sigma_{xy}(\cos^2\theta - \sin^2\theta) \tag{E.5}$$

$$= (\sigma_x^2 - \sigma_y^2) \sin 2\theta + 2\sigma_{xy} \cos 2\theta\,. \tag{E.6}$$

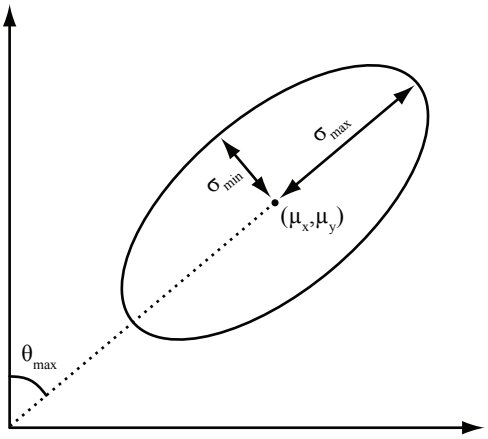


Figure E.1: An error ellipse.

This can be rearranged to give

$$\tan 2\theta = \frac{2\sigma_{xy}}{\sigma_y^2 - \sigma_x^2} \tag{E.7}$$

which has two solutions in the range $0^\circ - 180^\circ$ that are 90° apart . These solutions are the directions of the maximum and minimum standard deviations. If $\sigma_{xy} > 0$ then the value of θ that lies in the range $0^\circ - 90^\circ$ is the direction of the maximum deviation. Otherwise the direction of the maximum deviation is the solution in the range $90^\circ - 180^\circ$. The values of the maximum and minimum standard deviations are found by substituting the two solutions back into Equation E.4 and taking the square root of each result.

Let $\sigma_{\max}$ and $\sigma_{\min}$ be the maximum and minimum deviations with corresponding directions $\theta_{\max}$ and $\theta_{\min}$. The standard error ellipse given these values is drawn as shown in Figure E.1. The probability that the true position of the point actually lies within this ellipse is 39%. An error ellipse that contains the true position of the point with probability p can be obtained by multiplying the major and minor axis lengths by a scaling factor [102]. For 2-dimensional error ellipses the scaling factor is given by

$$k = \sqrt{-2\log_e(1-p)}\,. \tag{E.8}$$

For example the 95% error ellipse is obtained by calculating the standard error ellipse and multiplying the axis lengths by $k = 2.447$.

In 3-dimensional space it is possible to compute an error ellipsoid to represent the uncertainty of the position of a point [101]. Error ellipsoids can also be generalised to represent uncertainty in spaces that have more than three dimensions.

Appendix F

Glossary

AOA	Angle of arrival
ARW	Angle random walk
EKF	Extended Kalman filter
GBP	Great British pound
GNSS	Global navigation satellite system
IMU	Inertial measurement unit
INS	Inertial navigation system
KF	Kalman filter
KLD	Kullback-Leibler distance
LAN	Local area network
LED	Light emitting diode
MEMS	Micro-machined electromechanical systems
NTP	Network time protocol
OSI	Open system interconnect
PDR	Pedestrian dead reckoning
RF	Radio frequency
RFID	Radio frequency identification
RSSI	Received signal strength indication
SCAAT	Single constraint at a time tracking
SIR	Sampling importance resampling
TDOA	Time difference of arrival
TOA	Time of arrival
UMPC	Ultra mobile personal computer
UWB	Ultra wideband
VRW	Velocity random walk
ZVU	Zero velocity update

References

[1] M. Weiser. The computer for the 21st century. *Scientific American*, 265:66–75, 1991.

Cited on page: 1.

[2] R. Harle. *Maintaining world models in context-aware environments*. PhD thesis, University of Cambridge, Department of Engineering, 2004.

Cited on page: 1.

[3] M. Hazas, J. Scott, and J. Krumm. Location-aware computing comes of age. *Computer*, 37:95–97, 2004.

Cited on page: 2.

[4] S. Gleason and D. Gebre-Egziabher. *GNSS applications and methods*. Artech House, 2009.

Cited on pages: 2 and 9.

[5] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. Technical Report 92.1, Olivetti Research Ltdaddle, 1992.

Cited on pages: 2, 7, and 11.

[6] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *Computer*, 34:50–56, 2001.

Cited on pages: 2, 7, 12, 76, 185, and 188.

[7] M. Hazas and A. Ward. A novel broadband ultrasonic location system. In *Proceedings of the 4th international conference on Ubiquitous Computing (UbiComp 2002)*, pages 264–280, 2002.

Cited on pages: 2 and 12.

[8] N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 32–43, 2000.

Cited on pages: 2, 13, and 185.

[9] S. Gezici, Zhi Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *Signal Processing Magazine, IEEE*, 22(4):70–84, 2005.

Cited on page: 2.

[10] P. Bahl and V.N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the 19th Joint Conference of the IEEE Computer and Communications Societies (InfoCom 2000)*, volume 2, pages 775–784, 2000.

Cited on pages: 2, 15, and 142.

[11] W. Hui, H. Lenz, A. Szabo, J. Bamberger, and U.D. Hanebeck. WLAN-based pedestrian tracking using particle filters and low-cost MEMS sensors. In *Proceedings of the 4th Workshop on Positioning, Navigation and Communication (WPNC 2007)*, pages 1–7, 2007.

Cited on pages: 2, 15, and 23.

[12] M. Youssef and A. Agrawala. The Horus WLAN location determination system. In *Proceedings of the 3rd international Conference on Mobile Systems, Applications, and Services (MobiSys 2005)*, pages 205–218, 2005.

Cited on pages: 2, 15, and 142.

[13] D. Titterton and J. Weston. *Strapdown Inertial Navigation Technology*. The American Institute of Aeronautics and Astronautics, 2nd edition, 2004.

Cited on pages: 3, 20, 32, 42, 43, 46, 47, 48, 49, 50, 54, and 175.

[14] E. Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

Cited on pages: 3, 19, 24, 25, 33, 58, 60, and 76.

[15] P. Groves, G. Pulford, C. Mather, A. Littlefield, D. Nash, and M. Carter. Integrated pedestrian navigation using GNSS, MEMS IMU, magnetometer and baro-altimeter. In *The Navigation Conference and Exhibition (NAV 2007)*, 2007.

Cited on pages: 3, 23, and 24.

[16] S. Godha, G. Lachapelle, and M. E. Cannon. Integrated GPS/INS system for pedestrian navigation in a signal degraded environment. In *Proceedings of the 19th International Technical Meeting of the Institute of Navigation Satellite Division (ION-GNSS 2006)*, September 2006.

Cited on pages: 3 and 24.

[17] G. Welch and E. Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22:24–38, 2002.

Cited on pages: 7 and 32.

[18] P. Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech House, 2008.

Cited on pages: 7, 9, 19, 41, 44, 46, and 73.

[19] E. Foxlin, M. Harrington, and G. Pfeifer. Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set applications. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1998)*, pages 371–378, 1998.

Cited on pages: 7, 32, and 78.

[20] J. Hightower, B. Brumitt, and G. Borriello. The location stack: A layered model for location in ubiquitous computing. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (HotMobile 2002)*, pages 22–28, 2002.

Cited on page: 8.

[21] J. Hightower, D. Fox, and G. Borriello. The location stack. Technical Report 03-07-01, University of Washington, Department of Computer Science and Engineering, 2003.

Cited on pages: 8 and 25.

[22] A. Bensky. *Wireless positioning technologies and applications*. Artech House, 2008.

Cited on pages: 8, 10, and 14.

[23] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, 2003.

Cited on pages: 8, 26, and 28.

[24] J. Swann and D. Ludwig. Galileo: Benefits for location based services. *Journal of Global Positioning Systems*, 1(2):57–66, 2003.

Cited on page: 9.

[25] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Placelab: Device positioning using radio beacons in the wild. In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, 2005.

Cited on pages: 10 and 11.

[26] C. Drane, M. Macnaughtan, and C. Scott. Positioning GSM telephones. *IEEE Communications Magazine*, 36(4):46–59, 1998.

Cited on page: 10.

[27] G. Dedes and A. Dempster. Indoor GPS: Positioning challenges and opportunities. In *Proceedings of the 62nd IEEE Vehicular Technology Conference (VTC 2005-Fall)*, pages 412–415, 2005.

Cited on page: 11.

[28] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. In *Proceedings of the 5th International Conference on Mobile Computing and Networking (MobiCom 1999)*, pages 59–68, 1999.

Cited on pages: 12, 185, and 188.

[29] N.B. Priyantha. *The cricket indoor location system*. PhD thesis, Massachusetts Institute of Technology, 2005.

Cited on pages: 13 and 185.

[30] A. Smith, H. Balakrishnan, M. Goraczko, and N.B. Priyantha. Tracking moving devices with the cricket location system. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications and Services (MobiSys 2004)*, pages 190–202, 2004.

Cited on pages: 13 and 30.

[31] M. Hazas and A. Ward. A high performance privacy-oriented location system. In *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 216–223, 2003.

Cited on page: 13.

[32] M. Lott and I. Forkel. A multi-wall-and-floor model for indoor radio propagation. In *Proceedings of the 53rd IEEE Vehicular Technology Conference (VTC 2001-Spring)*, volume 1, pages 464–468, 2001.

Cited on page: 14.

[33] K. Kaemarungsi. Distribution of WLAN received signal strength indication for indoor location determination. In *Proceedings of the 1st International Symposium on Wireless Pervasive Computing (ISWPC 2006)*, pages 6–11, 2006.

Cited on pages: 15 and 149.

[34] P. Castro, P. Chiu, T. Kremenek, and R. Muntz. A probabilistic room location service for wireless networked environments. In *Proceedings of the 3rd International Conference on Ubiquitous Computing (UbiComp 2001)*, pages 18–34, 2001.

Cited on pages: 15 and 142.

[35] A. Haeberlen, E. Flannery, A.M. Ladd, A. Rudys, D.S. Wallach, and L.E. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the 10th International Conference on Mobile Computing and Networking (MobiCom 2004)*, pages 70–84, 2004.

Cited on pages: 15 and 142.

[36] Q. Cai and J. K. Aggarwal. Tracking human motion using multiple cameras. In *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR 1996)*, volume 3, pages 68–72, 1996.

Cited on page: 16.

[37] A.R. Dick and M.J. Brooks. A stochastic approach to tracking objects across multiple cameras. *Advances in Artificial Intelligence*, 3339:160–170, 2004.

Cited on page: 16.

[38] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for EasyLiving. In *Proceedings of the 3rd IEEE International Workshop on Visual Surveillance (VS 2000)*, page 3, 2000.

Cited on page: 16.

[39] S. Hay and R. Harle. Bluetooth tracking without discoverability. In *Proceedings of the 4th International Symposium on Location and Context Awareness (LoCA 2009)*, pages 120–137, 2009.

Cited on page: 16.

[40] A. Mandal, C.V. Lopes, T. Givargis, A. Haghighat, R. Jurdak, and P. Baldi. Beep: 3d indoor positioning using audible sound. In *Proceedings of the 2nd IEEE Consumer Communications and Networking Conference (CCNC 2005)*, pages 348–353, 2005.

Cited on page: 16.

[41] S. Patel, K. Truong, and G. Abowd. Powerline positioning: A practical sub-room-level indoor location system for domestic use. In *Proceedings of the 8th international conference on Ubiquitous computing (Ubicomp 2006)*, pages 441–458, 2006.

Cited on page: 16.

[42] E. Foxlin. Motion tracking requirements and technologies. *Handbook of Virtual Environments: Design, Implementation, and Applications*, pages 163–210, 2002.

Cited on pages: 20, 21, and 45.

[43] D. Cyganski, J. Duckworth, S. Makarow, W. Michalson, J. Orr, V. Amendolare, J. Coyne, G. Daempfling, J. Farmer, D. Holl, S. Kilkarni, H. Parikh, and B. Woodacre. WPI precision personnel locator system. In *Institute of Navigation, National Technical Meeting (ION-NTM 2007)*, 2007.

Cited on page: 22.

[44] D. Alvarez, R.C. Gonzalez, A. Lopez, and J.C. Alvarez. Comparison of step length estimators from wearable accelerometer devices. In *Proceedings of the 28th IEEE International Conference on Engineering in Medicine and Biology Society (EMBS 2006)*, pages 5964–5967, 2006.

Cited on page: 23.

[45] S. Y. Cho and C. G. Park. MEMS based pedestrian navigation system. In *The Journal of Navigation*, pages 135–163, 2006.

Cited on page: 23.

[46] S. Beauregard. A helmet-mounted pedestrian dead reckoning system. In *Proceedings of the 3rd International Forum on Applied Wearable Computing (IFAWC 2006)*, pages 79–89, 2006.

Cited on page: 23.

[47] P. Groves, G. Pulford, A. Littlefield, D. Nash, and C. Mather. Inertial navigation verses pedestrian dead reckoning: Optimizing the integration. In *Proceedings of the 20th International Technical Meeting of the Institute of Navigation Satellite Division (ION-GNSS 2007)*, 2007.

Cited on page: 23.

[48] Ulrich Steinhoff and Bernt Schiele. Dead reckoning from the pocket - an experimental study. In *Proceedings of the 8th International Conference on Pervasive Computing and Communications (PerCom 2010)*, 2010.

Cited on pages: 23, 24, and 164.

[49] H. Weinberg. Using the ADXL202 in pedometer and personal navigation applications. 2002.

Cited on page: 23.

[50] J. Kappi, J. Syrjarinne, and J. Saarinen. MEMS-IMU based pedestrian navigator for handheld devices. In *Proceedings of the 14th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION-GPS 2001)*, 2001.

Cited on page: 23.

[51] C. Mather, P. Groves, and M. Carter. A man motion navigation system using high sensitivity GPS, MEMS IMU and auxiliary sensors. In *Proceedings of the 19th International Technical Meeting of the Institute of Navigation Satellite Division (ION-GNSS 2006)*, 2006.

Cited on page: 24.

[52] D. Simon. *Optimal state estimation: Kalman, H_∞, and nonlinear approaches*. Wiley, 2006.

Cited on pages: 25 and 179.

[53] R. Brown and P. Hwang. *Introduction to random signals and applied Kalman filtering*. Wiley, 1997.

Cited on pages: 26, 27, 64, 179, 180, and 182.

[54] S.M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

Cited on pages: 27 and 28.

[55] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer, 1st edition, 2001.

Cited on pages: 29 and 92.

[56] J. Hol, T. Schön, and F. Gustafsson. On resampling algorithms for particle filters. In *Proceedings of the IEEE Nonlinear Statistical Signal Processing Workshop (NSSPW 2006)*, pages 79–82, 2006.

Cited on page: 29.

[57] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI 1999)*, pages 343–349, 1999.

Cited on pages: 29, 33, 36, 89, and 117.

[58] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research*, 22:985–1003, 2003.

Cited on pages: 29, 36, 119, and 122.

[59] G. Welch and G. Bishop. SCAAT: incremental tracking with incomplete information. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1997)*, pages 333–344, 1997.

Cited on page: 30.

[60] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST 1999)*, pages 1–10, 1999.

Cited on page: 30.

[61] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. High-performance wide-area optical tracking: The hiball tracking system. *Presence: Teleoperators and Virtual Environments*, 10(1):1–21, 2001.

Cited on page: 30.

[62] B. Scherzinger. Precise robust positioning with inertial/GPS RTK. In *Proceedings of the 13th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION-GPS 2000)*, 2000.

Cited on page: 32.

[63] K.A. Ornedo, R.S. Farnsworth and G.S. Sandhoo. GPS and radar aided inertial navigation system for missile system applications. In *Proceedings of the Position, Location and Navigation Symposium (PLANS 1998)*, pages 614–621, 1998.

Cited on page: 32.

[64] A. Brown and D. Sullivan. Precision kinematic alignment using a low-cost GPS/INS system. In *Proceedings of the 15th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION-GPS 2002)*, 2002.

Cited on page: 32.

[65] J. Hightower. *The location stack*. PhD thesis, University of Washington, Department of Computer Science and Engineering, 2004.

Cited on page: 32.

[66] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

Cited on pages: 33, 34, 35, 36, and 89.

[67] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1999)*, volume 2, pages 588–594, 1999.

Cited on page: 33.

[68] H.R. Everett. *Sensors for mobile robots: Theory and application*. AK Peters, 1995.

Cited on page: 34.

[69] K.S Chong and L. Kleeman. Accurate odometry and error modelling for a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2783–2788, 1997.

Cited on page: 34.

[70] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Position tracking with position probability grids. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots (EuroBot 1996)*, pages 2–9, 1996.

Cited on pages: 34 and 35.

[71] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. *Sequential Monte Carlo methods in practice*, pages 499–516, 2001.

Cited on pages: 34 and 36.

[72] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996)*, volume 2, pages 896–901, 1996.

Cited on page: 35.

[73] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proceedings of the 15th International Joint Conference on Artifical intelligence (IJCAI 1997)*, pages 1346–1352, 1997.

Cited on page: 38.

[74] iSuppli. MEMS January 2010 market brief, 2010.

Cited on page: 38.

[75] B. Krach and P. Robertson. Integration of foot-mounted inertial sensors into a Bayesian location estimation framework. In *Proceedings of the 5th Workshop on Positioning, Navigation and Communication (WPNC 2008)*, pages 55–61, 2008.

Cited on page: 39.

[76] B. Krach and P. Robertson. Cascaded estimation architecture for integration of foot-mounted inertial sensors. *Proceedings of the Position, Location and Navigation Symposium (PLANS 2008)*, pages 112–119, 2008.

Cited on page: 39.

[77] Widyawan, M. Klepal, and S. Beauregard. A backtracking particle filter for fusing building plans with PDF displacement estimates. In *Proceedings of the 5th Workshop on Positioning, Navigation and Communication (WPNC 2008)*, pages 207–212, 2008.

Cited on pages: 39 and 40.

[78] S. Beauregard, Widyawan, and M. Klepal. Indoor PDR performance enhancement using minimal map information and particle filters. In *Proceedings of the Position, Location and Navigation Symposium (PLANS 2008)*, pages 141–147, 2008.

Cited on page: 39.

[79] O. Woodman. An introduction to inertial navigation. Technical Report 696, University of Cambridge, Computer Laboratory, 2007.

Cited on pages: 46 and 171.

[80] E. Foxlin. Inertial head-tracker sensor fusion by a complimentary separate-bias Kalman filter. In *Proceedings of the Virtual Reality Annual International Symposium (VRAIS 1996)*, pages 185–194, 1996.

Cited on page: 60.

[81] D. Roetenberg, H. Luinge, and P. Veltink. Inertial and magnetic sensing of human movement near ferromagnetic materials. In *Proceedings of the the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003)*, pages 268–269, 2003.

Cited on page: 85.

[82] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

Cited on page: 92.

[83] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA 2005)*, pages 64–69, 2005.

Cited on page: 96.

[84] G. Watson. *Statistics on spheres*. Wiley, 1983.

Cited on page: 105.

[85] L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz. Voronoi tracking: Location estimation using sparse and noisy sensor data. In *Proceedings. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 723–728, 2003.

Cited on pages: 112 and 135.

[86] B. Eissfeller, D. Gansch, S. Muller, and Teuber A. Indoor positioning using wireless LAN radio signals. In *Proceedings of the 17th International Technical Meeting of the Institute of Navigation Satellite Division (ION-GNSS 2004)*, pages 1936–1947, September 2004.

Cited on page: 112.

[87] D. Fox. KLD-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*, pages 713–720, 2001.

Cited on pages: 115, 117, 118, and 122.

[88] D. Ashbrook and T. Starner. Learning significant locations and predicting user movement with GPS. In *Proceedings of the 6th IEEE International Symposium on Wearable Computers (ISWC 2002)*, pages 101–108, 2002.

Cited on page: 125.

[89] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

Cited on page: 125.

[90] A. Milstein, J.N. Sánchez, and E.T. Williamson. Robust global localization using clustered particle filtering. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002)*, pages 581–586, 2002.

Cited on page: 127.

[91] A. Rice and O. Woodman. Crowd-sourcing world models with OpenRoomMap. In *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications (PerCom 2010), Work in Progress*, 2010.

Cited on page: 130.

[92] M. Haklay and P. Weber. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

Cited on page: 130.

[93] M. Held. VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Computational Geometry*, 18(2):95–123, 2001.

Cited on page: 135.

[94] M. Ocana, L. M. Bergasa, M. A. Sotelo, J. Nuevo, and R. Flores. Indoor robot localization system using WiFi signal measure and minimizing calibration effort. In *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE 2005)*, pages 1545–1550, 2005.

Cited on page: 142.

[95] M. Lee, H. Yang, D. Han, and C. Yu. Crowdsourced radiomap for room-level place recognition in urban environment. In *Proceedings of the IEEE Workshop on Smart Environments (SmartE 2010)*, 2010.

Cited on page: 142.

[96] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1977)*, 11(2):214–222, 1977.

Cited on page: 144.

[97] O. Woodman and R. Harle. RF-Based initialisation for inertial pedestrian tracking. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive 2009)*, pages 238–255, 2009.

Cited on page: 152.

[98] W. Stockwell. *Bias Stability Measurement : Allan Variance*. http://www.xbow.com.

Cited on page: 169.

[99] IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros, Annex C. *IEEE Standard 952-1997*, 1998.

Cited on page: 170.

[100] A. Glassner. *Graphics Gems*, volume 1. Academic Press, 1990.

Cited on page: 176.

[101] W. Schofield and M. Breach. *Engineering Surveying*, chapter 7. Elsevier, 6th edition, 2007.

Cited on pages: 191 and 192.

[102] B. Hofmann-Wellenhof, K. Legat, and M. Wieser. *Navigation: Principles of positioning and guidance*, chapter 3, pages 46–48. Springer, 2003.

Cited on page: 192.