FILTERING AND TRACKING FOR A PEDESTRIAN DEAD-RECKONING SYSTEM

by

Surat Kwanmuang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2015

Doctoral Committee:

Professor A. Galip Ulsoy, Co-Chair
Associate Professor Edwin Olson, Co-Chair
Professor Johann Borenstein
Professor David K. Wehe

Localization is all about being lost and found.

To those who are my guiding stars, in the vast universe of life.

ACKNOWLEDGEMENTS

Throughout the time that I have been working on this dissertation, like all the matter in life, there were up and down moments. Without help and support from these people, I would never have come this far. I would like to express my sincerest gratitude to everyone that has made this thesis possible.

I would like to express my deepest appreciation to both of my co-chairs, Professor Galip Ulsoy and Professor Edwin Olson for their patient and gracious support. This dissertation would have never materialized without their encouragement and guidance.

I also would like to thank Professor Johann Borenstein for his guidance and support during my precious time in the Mobile Robotics Laboratory, which was the starting point of my interest in the inertial navigation system and the beginning of this dissertation.

In addition, I would like to thank my cognate member, Professor David Wehe for his kindness in accommodating his time during his sabbatical to make this dissertation defense possible.

Moreover, a thank you to Professor Roland Graf who has always supported and encouraged me to finish this dissertation.

On a more personal note, I would like to thank my parents, Surachai and Shauladda Kwanmuang, for their unconditional love and support. I also would like to thank Sanikan Saingam for her support and encouragement during the preparation of this thesis and the Hiser family for their support during my last few months in the US.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

CRP Classical Rodrigues Parameters

DCM Direction Cosine Matrix

EKF Extended Kalman Filter

GPS Global Positioning System

GRP Generalized Rodrigues Parameters

HDE Heuristic Drift Elimination

IMU Inertial Measurement Unit

KF Kalman Filter

MEMS Micro-Electro-Mechanical Systems

ML Maximum-Likelihood

MRP Modified Rodrigues Parameters

PDR Pedestrian Dead-Reckoning

SGD Stochastic Gradient Descent

SLAM Simultaneous Localization And Mapping

UKF Unscented Kalman Filter

ZARU Zero Attitude Rate Update

ZUPT Zero Velocity Update

ABSTRACT

Filtering and Tracking for a Pedestrian Dead-Reckoning System

by

Surat Kwanmuang

Co-Chairs: A. Galip Ulsoy and Edwin Olson

This thesis proposes a leader-follower system in which a robot, equipped with relatively sophisticated sensors, tracks and follows a human whose equipped with a low-fidelity odometry sensor called a Pedestrian Dead-Reckoning (PDR) device. Such a system is useful for "pack mule" applications, where the robot carries heavy loads for the humans. The proposed system is not dependent upon GPS, which can be jammed or obstructed.

This human-following capability is made possible due to several novel contributions. First, we perform an in-depth analysis of our Pedestrian Dead-Reckoning (PDR) system with the Unscented Kalman Filter (UKF) and models of varying complexity. We propose an extension that limits elevation errors, and show that our proposed method reduces errors by 63% compared to a baseline method. We also propose a method for integrating magnetometers into the PDR framework, which automatically and opportunistically calibrates for hard/soft-iron effects and sensor misalignments. In a series of large-scale experiments, we show that this system achieves positional errors of less than 1.9% of the distance traveled.

Finally, we propose methods that allow a robot to use LIDAR data to improve the accuracy of the robot's estimate of the humans trajectory. These methods include: 1) a particle filter method and 2) two multi-hypothesis maximum-likelihood approaches based on stochastic gradient descent optimization. We show that the proposed approaches are able to track human trajectories in several synthetic and real-world datasets.

CHAPTER I

Introduction

1.1 Motivation

Today, the majority of interactive robots are pre-programmed or tele-operated robots; researchers, however, have been looking into capable and fully autonomous robots. There are many contemporary applications where a human would retain authority and the robots only require partially autonomy. One such example of partially-autonomous behavior would be having the robot follow a human leader. This application is also known as "human leader and robot follower" or "leader-follower". In a typical application, the human leader would be paired with a robot follower without the human needing to actually teleoperate the robot. This behavior has many potential applications for military, transportation, and personal usage. In the military, these robots could be used to carry massive amounts of supplies during a mission without having to divert limited manpower (Figure 1.1). This same behavior can be applied directly to robotic convoys used in various transportation businesses. Meanwhile, in the realm of personal usage, robots could provide smart baggage services by following a particular individual through an airport (Figure 1.2), or provide similar smart cart services in shopping centers.

In order to enable any of these behaviors, a robot must be able to track the position of the human leader. Although various sensors can be used for tracking, there

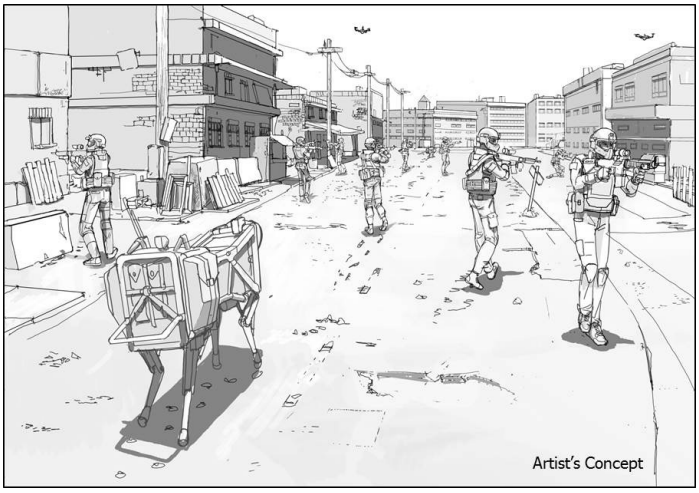

Figure 1.1: Artist's rendering of the Squad-X Core Technology concept. The concept envisions the future of a team of soldiers working together with a robot follower. (Image courtesy of DARPA)

Figure 1.2: Hop! following suitcase (*García*, 2012). The suitcase utilizes bluetooth receivers to track a Bluetooth beacon from the leader's phone.

are limitations. Line-of-sight measurements such as cameras or beacons have field-of-view and range limitations. Global Positioning System (GPS) position tracking is only available for outdoor environments and can be jammed or obstructed. Additionally, sensors that detect the leader's surroundings and use the data to match with the robot's surroundings—such as a laser scanner—are impractical to mount onto a human. As a result, we need a method that is less intrusive for tracking a human leader.

1.2 Purpose and Scope

In this thesis, we propose a new leader-follower approach that utilizes a Pedestrian Dead-Reckoning (PDR) system (*Ojeda and Borenstein*, 2007a; *Borenstein et al.*, 2009) on the human leader and a tracking algorithm on the robot follower. Our key idea is to equip the robot with relatively high-fidelity mapping sensors, and to use observations of the structured environment to constrain the path of the human.

Position output from the PDR system is not perfect due to performance limitations of the Inertial Measurement Unit (IMU) itself. This results in position errors, especially elevation and heading error, which limit its usefulness in some applications.

This error may also lead to ambiguity in identifying the exact path taken by the human leader. To solve this problem, the robot needs a tracking method to combine leader trajectory data with the robot's own map.

This thesis consists of two parts. The first focuses on estimating the position of the human leader. The second describes how the robot combines this data with the robot's map, further improve the estimate of the leader's trajectory.

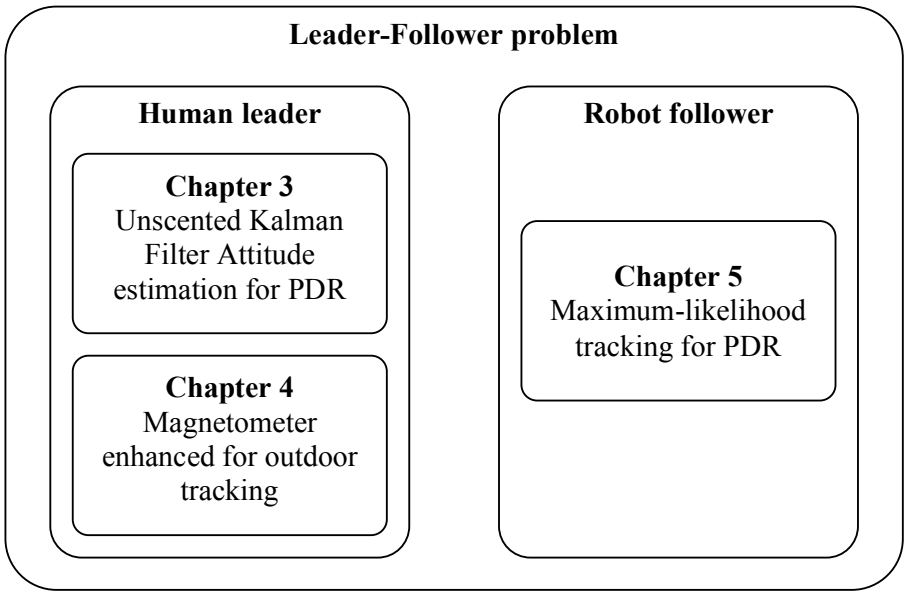

Figure 1.3: Organization of chapters within this thesis.

Figure 1.3 shows the organization of the main chapters within this thesis. Chapter II provides the background necessary for the following chapters. Chapter III explores the application of Unscented Kalman Filter (UKF) to the PDR problem in order to improve position estimation in general. Chapter IV focuses on adding magnetometers to improve position estimation in outdoor environments.

Chapter V presents a new method for tracking leader position that implements a Maximum-Likelihood (ML) solution and Stochastic Gradient Descent (SGD) technique. Finally, Chapter VI summarizes the dissertation and research concluding and discusses future research work.

1.3 Original Contributions

The contributions of this thesis include:

- An in-depth analysis of attitude and position estimation using Unscented Kalman Filter (UKF) applied to PDR problem and comprehensive comparisons of various design-choices such as model complexity and noise model
- A practical and robust calibration method for incorporating magnetometers into the PDR system to improve position estimation in outdoor environments
- Novel tracking algorithms that fuse both PDR data from the human and LIDAR data from the following robot, including methods based on particle filters and Stochastic Gradient Descent (SGD)
- A multi-hypothesis tracking algorithm based on Stochastic Gradient Descent (SGD) that is both fast and robust

CHAPTER II

Background

2.1 The Pedestrian Dead-Reckoning (PDR) system

The goal of the Pedestrian Dead-Reckoning (PDR) project (*Ojeda and Borenstein*, 2007a; *Borenstein et al.*, 2009) was to track the position of a human subject in real-time. The PDR system uses a foot-mounted Inertial Measurement Unit (IMU), which includes a three-axis gyroscope, a three-axis accelerometer, and a three-axis magnetometer. The IMU is strapped to the side or embedded in the heel of the user's boot, as shown in Figure 2.1. The side-mounted IMU can be transferred easily between different users while the in-heel version better protects the IMU from damage and cannot be dislocated easily. The PDR system's computations are performed on a portable computer that is located inside a belt pack, together with batteries and support electronics. The two Micro-Electro-Mechanical Systems (MEMS)-based IMUs used in the PDR system are Memsense nIMU and Intersense Navchip.

An IMU-based position estimation system combines two functions: the estimation of distance traveled and the estimation of heading. In the PDR system, the accuracy of both components is predominantly affected by bias drift (or just "drift"). This is especially true if a relatively low-performance MEMS-based IMU is used. Drift rates for both accelerometers and gyroscopes in a MEMS-based IMU are several orders of magnitude higher than what is found in high-grade aviation IMUs. Of course, the cost

Figure 2.1: The foot-mounted IMU of the PDR system has two mounting options. (a) Side-mounted IMU (the IMU itself is covered by the beige-colored thermal insulation). (b) In-heel IMU with temperature controlled, shock resistant housing

of high-grade IMUs is also several orders of magnitude higher than that of the IMUs in the PDR system. In addition to cost, size is a critical constraint in the PDR system. In order to embed the IMU in the heel of a regular firefighter or military-style boot, the device must be very small. Because of this size-limitation, the only currently suitable IMU technology is Micro-Electro-Mechanical Systems (MEMS). Another limiting factor is the fact that the peak accelerations and rates of turns made by a foot, even at normal walking speed, are significantly higher than those found on the torso of a person. This limits the choice of suitable IMUs to less than a handful of models that offer the large dynamic range required by that particular IMU location.

A foot-mounted IMU makes wiring more difficult, requires greater dynamic ranges of the IMU, and may add further difficulties in heading estimation. Nevertheless, there is still a compelling reason for choosing this mounting location in spite of these drawbacks: it offers a way to estimate the drift of the accelerometers and subtract this drift from subsequent readings almost as frequently as once every second.

This method of resetting drift is called Zero Velocity Update (ZUPT), and it is possible with a foot-mounted IMU because the instrumented foot has zero velocity

during the brief moment when the foot is fully in contact with the floor. We call this moment of zero foot velocity, "the footfall". Human walking allows ZUPTs to be performed naturally as frequently as once every other step. Because ZUPTs can be applied so frequently, errors due to accelerometer drift can be eliminated almost entirely.

Since the PDR system estimates the distance and direction of the foot motion itself, there is no need to calibrate the system for each user. Also, the PDR system works equally well when users walk sideways, backwards, or in uncommon gaits. A more complete discussion of this function of the PDR system is provided in *Ojeda and Borenstein* (2007a) and *Borenstein et al.* (2009).

2.2 Unscented Kalman Filtering

The Unscented Kalman Filter (UKF) was developed by Julier Uhlmann and Durrant-Whyte (*Julier et al.*, 1995). UKF is a filter in the Kalman filter family that has several advantages over the original Kalman Filter (KF) or the Extended Kalman Filter (EKF). For example, it does not requite Jacobians or explicit linearization of non-linear systems, and it often produces better mean and variance estimates in non-linear problems.

A discrete-time, non-linear, system model with additive noise can be described as:

$$x_k = f(x_{k-1}, u_{k-1}) + G_k w_k \tag{2.1}$$

$$\tilde{y}_k = h(x_k) + v_k \tag{2.2}$$

where x_k is $n \times 1$ state vector at the k^{th} time-step and $\tilde{y}_k$ is $m \times 1$ measurement vector and u_k is a system input. A process noise w_k and measurement noise v_k are zero-mean white noise with covariance Q_k and R_k, respectively. A function $f(x_{k-1}, u_{k-1})$ is a non-linear state propagation function and $h(x_k)$ is a non-linear

observation function.

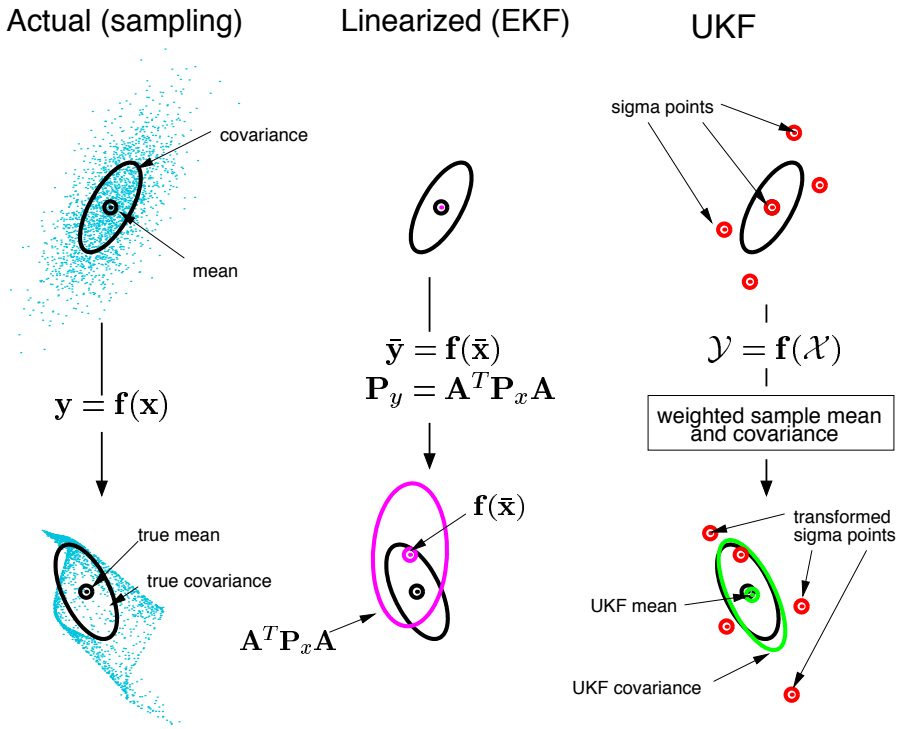

Figure 2.2: Mean and covariance propagation. (Left) Actual, (Middle) EKF and (Right) UKF (*Van der Merwe et al.*, 2001)

The UKF has a unique method of propagating covariance through a system model. In contrast, with a linear KF or EKF filter the covariance P is propagated through a linear/linearized system matrix F in the form of:

$$\bar{P}_k = F_k P_{k-1} F_k^T + Q_k \tag{2.3}$$

The above method has a limitation in a non-linear system in that the covariance is propagated through a linear approximation version of the system model F_k. For a highly non-linear system, this method may yield larger approximation errors.

UKF, on the other hand, utilizes a set of points called *sigma points* which are generated from the current system state and covariance. These sigma points are strategically chosen to be located at the mean and symmetrically along the main

axes of the covariance. For an n-dimensional Gaussian distribution with mean μ and covariance Σ, the $2n+1$ sigma points can be generated from:

$$X(0) = \mu \tag{2.4}$$

$$X(i) = \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \qquad \text{for } i = 1, ..., n \tag{2.5}$$

$$X(i) = \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \qquad \text{for } i = n+1, ..., 2n \tag{2.6}$$

where $\left(\sqrt{(n+\lambda)\Sigma}\right)_i$ represents the i^{th} column of the matrix $\sqrt{(n+\lambda)\Sigma}$.

The scalar $\lambda = \alpha^2(n+\kappa) - n$ indicates how far the spread of the sigma points vary from the mean. Normally the parameter α is in the range of $(0, 1)$ to control the spread and $\kappa \geq 0$ to guarantee positive definite-ness of the covariance matrix. Moreover, each sigma point has two weights associated with it: w_m for recovering mean and w_c for recovering covariance. They are defined by:

$$w_m(0) = \frac{\lambda}{n+\lambda} \tag{2.7}$$

$$w_c(0) = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{2.8}$$

$$w_m(i) = w_c(i) = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1, ..., 2n \tag{2.9}$$

The parameter $\beta \geq 0$ incorporates knowledge of the higher order moments of the distribution. For an exact Gaussian distribution, $\beta = 2$ is optimal.

We can see that the process of generating sigma points involves taking a square root of a matrix $(n+\lambda)\Sigma$. This can be implemented using Cholesky decomposition or LDL decomposition (*Guennebaud and Jacob*, 2010). Since the computational complexity of Cholesky decomposition is $O(n^3)$ (*Trefethen and Bau*, 1997), the size of matrix Σ has substantial effect on computational speed.

The mean μ and covariance Σ can be extracted from the sigma points using the

following:

$$\mu' = \sum_{i=0}^{2n} w_m(i) X(i) \tag{2.10}$$

$$\Sigma' = \sum_{i=0}^{2n} w_c(i) \left(X(i) - \mu'\right) \left(X(i) - \mu'\right)^T \tag{2.11}$$

In summary, the algorithm for the Unscented Kalman Filter (UKF) is described in Algorithm 2.1.

Algorithm 2.1 UKF with additive noise model (*Van der Merwe et al.*, 2001)

Initialize $\hat{x}_0$ and P_0 For all k, calculate sigma points, $\gamma = \sqrt{n + \lambda}$

$$X_{k-1} = \left[\hat{x}_{k-1} \quad , \quad \hat{x}_{k-1} + \gamma\sqrt{P_{k-1}} \quad , \quad \hat{x}_{k-1} - \gamma\sqrt{P_{k-1}}\right] \tag{2.12}$$

Time-update step for each sigma point $X_{k-1}(i)$, $i = 0, ..., 2n$

$$X_k(i)^- = F(X_{k-1}(i), u_{k-1}) \tag{2.13}$$

Recalculate norm and covariance of state variables

$$\hat{x}_k^- = \sum_{i=0}^{2n} w_m(i) X_k^-(i) \tag{2.14}$$

$$P_k^- = \sum_{i=0}^{2n} w_c(i) \left(X_{k-1}^-(i) - \hat{x}_k^-\right) \left(X_k^-(i) - \hat{x}_k^-\right)^T + Q \tag{2.15}$$

Measurement-update step for each sigma point

$$Z_k(i) = H(X_k^-(i)) \tag{2.16}$$

$$\hat{z}_k^- = \sum_{i=0}^{2n} w_m(i) Z_k(i) \tag{2.17}$$

$$P_{zz} = \sum_{i=0}^{2n} w_c(i) \left(Z_{k-1}(i) - \hat{z}_k^-\right) \left(Z_k(i) - \hat{z}_k^-\right)^T + R \tag{2.18}$$

$$P_{xz} = \sum_{i=0}^{2n} w_c(i) \left(X_{k-1}^-(i) - \hat{x}_k^-\right) \left(Z_k(i) - \hat{z}_k^-\right)^T \tag{2.19}$$

$$K = P_{xz} \cdot P_{zz}^{-1} \tag{2.20}$$

$$\hat{x}_k = \hat{x}_k^- + K(z - \hat{z}_k^-) \tag{2.21}$$

$$P_k = P_k^- - K P_{zz} K^T \tag{2.22}$$

In equation (2.1), we assume an additive noise model. In more complex system model, such as highly non-linear system, the noise model may not be easily linearized and separated into a linear additive term. One advantage of the UKF is that it is able to incorporate noise in a non-additive fashion. However, the number of sigma-points must typically be increased to preserve accuracy.

The system model with a non-additive noise can be written as:

$$x_k = f(x_{k-1}, u_{k-1}, w_k) \tag{2.23}$$

$$\tilde{y}_k = h(x_k, v_k) \tag{2.24}$$

A summary of the non-additive version of the UKF is shown in Algorithm 2.2. It can be seen that the process noise and measurement noise are augmented into the state (eq. 2.25) and covariance matrix (eq. 2.26), thereby increasing the number of states and sigma points. By directly incorporating noise into the model, the process noise matrix and measurement noise are not required as in the additive noise version.

Algorithm 2.2 UKF with non-additive noise model (*Van der Merwe et al.*, 2001)

Initialize $\hat{x}_0$ and P_0

States variable and covariance matrix are augmented with noise.

$$\hat{x}_0^a = \begin{bmatrix} \hat{x}_0^T & 0 & 0 \end{bmatrix}^T \tag{2.25}$$

$$P_0^a = \begin{bmatrix} P_0 & 0 & 0 \\ 0 & Q & 0 \\ 0 & 0 & R \end{bmatrix} \tag{2.26}$$

For all k, calculate sigma points, $\gamma = \sqrt{n + \lambda}$

$$X_{k-1}^a = \begin{bmatrix} \hat{x}_{k-1}^a & \hat{x}_{k-1}^a + \gamma\sqrt{P_{k-1}^a} & \hat{x}_{k-1}^a - \gamma\sqrt{P_{k-1}^a} \end{bmatrix} \tag{2.27}$$

$$= \begin{bmatrix} (X_{k-1}^x)^T & (X_{k-1}^Q)^T & (X_{k-1}^R)^T \end{bmatrix}^T \tag{2.28}$$

Time-update step for each sigma point $X_{k-1}^a(i)$, $i = 0, ..., 2n$

$$X_k(i)^{x-} = F(X_{k-1}^x(i), u_{k-1}, X_{k-1}^Q(i)) \tag{2.29}$$

Recalculate norm and covariance of state variables

$$\hat{x}_k^- = \sum_{i=0}^{2n} w_m(i) X_k^{x-}(i) \tag{2.30}$$

$$P_k^- = \sum_{i=0}^{2n} w_c(i) \left(X_{k-1}^{x-}(i) - \hat{x}_k^-\right) \left(X_k^{x-}(i) - \hat{x}_k^-\right)^T \tag{2.31}$$

Measurement-update step for each sigma point

$$Z_k(i) = H(X_k^{x-}(i), X_{k-1}^R(i)) \tag{2.32}$$

$$\hat{z}_k^- = \sum_{i=0}^{2n} w_m(i) Z_k(i) \tag{2.33}$$

$$P_{zz} = \sum_{i=0}^{2n} w_c(i) \left(Z_{k-1}(i) - \hat{z}_k^-\right) \left(Z_k(i) - \hat{z}_k^-\right)^T \tag{2.34}$$

$$P_{xz} = \sum_{i=0}^{2n} w_c(i) \left(X_{k-1}^-(i) - \hat{x}_k^-\right) \left(Z_k(i) - \hat{z}_k^-\right)^T \tag{2.35}$$

$$K = P_{xz} \cdot P_{zz}^{-1} \tag{2.36}$$

$$\hat{x}_k = \hat{x}_k^- + K(z - \hat{z}_k^-) \tag{2.37}$$

$$P_k = P_k^- - K P_{zz} K^T \tag{2.38}$$

2.3 Attitude Representations

Many mathematical representations can be used to define the attitude of a body with respect to a reference frame. Each representation requires a different number of parameters to describe an attitude and has different advantages and disadvantages.

First, the Direction Cosine Matrix (DCM) is a 3×3 matrix in which each column represents unit vectors in body axes projected along the reference axes. The DCM requires 9 parameters, one for each element in the matrix. Since the DCM is also a rotation matrix, all parameters also need to be satisfied for the rotation matrix constraints: $R^T = R^{-1}$ and $det(R) = 1$. These constraints and the relatively high number of parameters lead to difficulty in using the parameters directly in state estimation, despite the fact that the DCM does not have any mathematical singularity and the operations of DCMs are linear (*Markley*, 1978).

Second, Euler angles are successive transformations of one coordinate frame to another defined by three successive rotations. The Euler angle requires only three parameters as angles for each rotation. It is also possible to use Euler angles directly in state estimation, though care must be taken that they do have a mathematical singularity called the "gimbal lock" which is when the rotating axis of one rotation aligns with the axis of another (*Lefferts et al.*, 1982). This reduces the dimension space of attitude representation to two, because two rotations in the same axis can be considered one rotation.

Finally, the quaternion is a transformation of one coordinate to another through the description of a single rotation about a vector defined in a reference frame. The parameters of a quaternion are four numbers: three numbers to represent the vector ($\hat{e}$) and one number to specify the angle of rotation (Φ).

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T \tag{2.39}$$

$$q = \begin{bmatrix} e_x sin(\Phi/2) \\ e_y sin(\Phi/2) \\ e_z sin(\Phi/2) \\ cos(\Phi/2) \end{bmatrix} \tag{2.40}$$

$$q = \begin{bmatrix} \hat{e} sin(\Phi/2) \\ cos(\Phi/2) \end{bmatrix} \tag{2.41}$$

Though the quaternion multiplication is bilinear (*Gallier*, 2011) and do not involve a mathematical singularity, the quaternion still has to satisfy the constraint $q^T q = 1$. In some literature, the scalar part ($cos(\Phi/2)$) is in the first element instead of in the last as in the definition above, which may cause some confusion when it comes to implementing the quaternion.

Due to its compactness and lack of singularity, the quaternion has been chosen to be the attitude representation for the PDR system.

CHAPTER III

Filtering of a Pedestrian Dead-Reckoning System

3.1 Introduction

During the development of the Pedestrian Dead-Reckoning (PDR) system for firefighter applications, we found that the position output had a large elevation error. We mitigated this error by using a pair of barometric pressure sensors, one on the PDR system and the other on a fixed base station. The elevation can be calculated from the different atmospheric pressures between the two sensors. While this method works very well in most situations, it cannot work reliably in firefighter application, because the atmospheric pressure inside a burning building fluctuates highly due to the fire, thus rendering this method unusable. Unable to enlist the help of barometric pressure sensors, we decided to improve the performance of position estimation using inertial sensors only. After investigation, we suspected that the elevation error might be a result of attitude error. This can be observed in cases where a subject walks on a level floor but the elevation output looks like they are walking uphill or downhill. Though this kind of attitude error is small, it can significantly affect elevation estimation for long walks.

This chapter describes the construction of attitude and position estimation using the Unscented Kalman Filter (UKF) for the PDR application. This chapter will be useful for those who are interested in developing a PDR system, as it is a summary

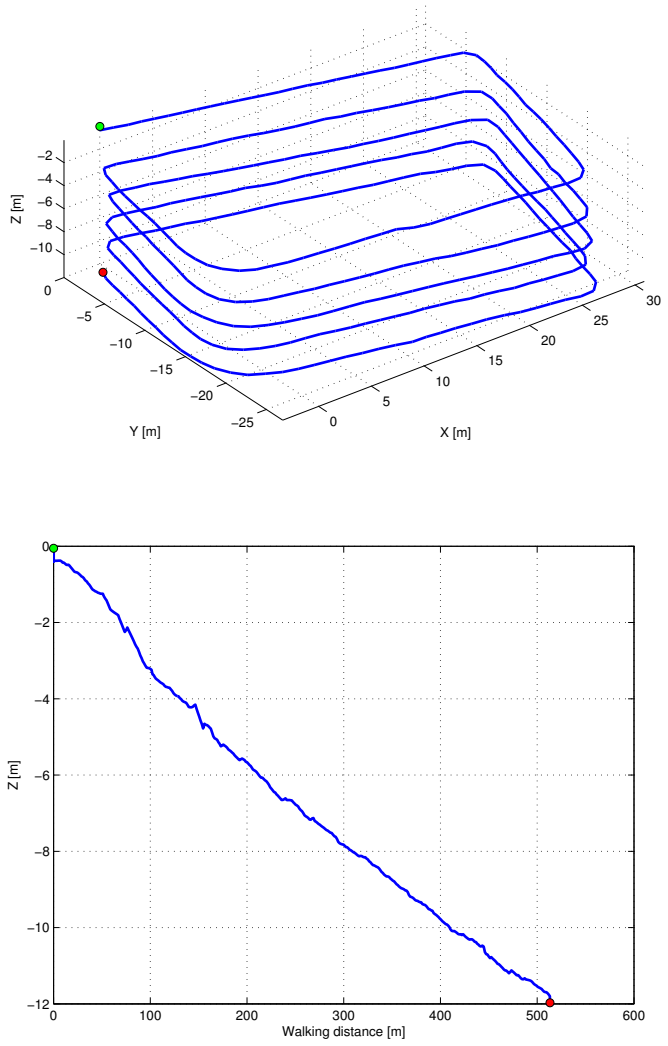

Figure 3.1: A 3-dimensional trajectory output from the PDR system (top) of a subject walking along a level rectangular path. Bottom figure shows elevation error of the output as a function of walking distance.

of all major steps needed.

This chapter is divided into two parts: attitude estimation and velocity/position estimation. For attitude estimation, a gyroscope model is introduced and quaternion attitude propagation steps are derived. Likewise, for velocity/position estimation, the accelerometer model and velocity/position model are derived.

In addition, we have constructed a system model with two types of noise models and included a discussion of measurements that can be used to the system. Moreover, the method of modeling actual sensor noise models is also introduced for the preparation of working with actual experiment data, followed by a comparison of characteristics of the two Inertial Measurement Units (IMUs) that were used. Towards the end of the chapter, experimental results from a large dataset are provided and discussed.

The main contributions of this chapter are:

- A summary of methods for estimating attitude and velocity/position using measurements from the IMU
- An analysis of attitude/position estimation in an Unscented Kalman Filter (UKF) framework with various model complexities and noise models
- An evaluation of our experimental dataset that demonstrates the performance of each implementation and their comparison

3.2 Prior work

There are several approaches to estimate the position of a person without the use of a Global Positioning System (GPS). The most basic approach extends the basic function of a pedometer by counting the number of steps taken together with a constant pre-defined step length to calculate walking distance. *Judd* (1997) also incorporates a digital compass for walking direction to calculate the 2D position of

the subject. Since step length can vary with walking speed, *Mezentsev et al.* (2005) assumed step length to be a random walk process. *Weinberg* (2002) and *Randell et al.* (2003) tried to distinguish walking speed from acceleration measured and inferred step length from estimated walking speed. *Beauregard and Haas* (2006), on the other hand, used a neural network to estimate step length. These approaches all have errors resulting from the simplification of step length, as well as directional error from magnetometer disturbances.

More recently, since the development of inertial sensors such as gyroscopes and accelerometers has become increasingly sophisticated and miniaturized, strap-down inertial navigation has been adopted in many mobile devices. Some systems use off-the-shelf inertial navigation systems such as Xsens MTi which offers a basic attitude estimate (*Beauregard*, 2009; *Jimenez et al.*, 2009). Other research systems implement both attitude and position estimation algorithms using complimentary filters (*Ojeda and Borenstein*, 2007a) or Extended Kalman Filter (EKF) (*Foxlin*, 2005). The closest prior research to our work is *Zampella et al.* (2012), particularly in terms of their use of UKF for the PDR application. Their work, however, utilizes Euler angles as state variables while our work uses quaternions, which do not suffer from gimbal lock (Section 2.3).

3.3 Attitude estimation

The attitude estimation in the current PDR system (*Ojeda and Borenstein*, 2007a) uses a complementary filter and does not account for noise and bias drift from inertial sensors. To improve attitude estimation, we developed a new estimation method with a more detailed sensor model by expanding the framework described in *Crassidis and Markley* (2003) and compared the performance against current PDR systems.

3.3.1 Gyroscope model

A gyroscope can be used to measure a rotational rate around a sensitive axis. There are several types of gyroscopes available with various performances and sizes (*Titterton and Weston*, 2004). For the PDR system, MEMS gyroscopes were chosen due to their compactness, which allowed installation in a boot. The measured rotation rates from a 3-axis gyroscope, as a function of true rotation rate and sensor imperfections, can be written as (*Titterton and Weston*, 2004):

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_m = \begin{bmatrix} 1+S_x & M_{xy} & M_{xz} \\ M_{yz} & 1+S_y & M_{yz} \\ M_{zx} & M_{zy} & 1+S_z \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_t + \begin{bmatrix} G_{xx} & G_{xy} & G_{xz} \\ G_{yx} & G_{yy} & G_{yz} \\ G_{zx} & G_{zy} & G_{zz} \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_g + \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \tag{3.1}$$

$$\omega_m = S \cdot \omega_t + M \cdot \omega_t + G \cdot a + b_g + n_\omega \tag{3.2}$$

where ω_m is rate measurement from the sensor and ω_t is true rotation rate. S represents the scale factor coefficients in each axis. M represents cross-coupling coefficients and G represents sensitivity to linear acceleration a. The b's are biases and n_ω are measurement noises which are assumed to be zero-mean Gaussian noise $N(0, \sigma_\omega^2)$.

The scale factor and cross-coupling coefficients are pre-calibrated from the factory, and the sensitivity to linear acceleration coefficients can be calibrated when the PDR is assembled. We use the measurement function below:

$$\omega_m \equiv \omega_t + b_g + n_\omega \tag{3.3}$$

The gyro bias b_g is assumed to be a random walk process:

$$\dot{b}_g = n_{bg} \tag{3.4}$$

where n_{bg} is assumed to be zero-mean Gaussian noise $N(0, \sigma_{bg}^2)$

We will discuss in more detail regarding the parameterization of these sensor imperfection variables in section 3.7.

3.3.2 Attitude propagation

The angular rates from gyroscopes can be integrated with respect to time to compute current orientation. Since we are using quaternions as our attitude representation, the integration has to be performed in quaternion space. By doing so, the attitude error from conversion can be avoided by working directly in quaternion space. From *Trawny and Roumeliotis* (2005), the differential quaternion is a function of the current quaternion (q_t) and angular rates in the body axis (ω):

$$\dot{q}_t = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} q_t \tag{3.5}$$

$$\dot{q}_t = \frac{1}{2} \begin{bmatrix} -[\omega\times] & \omega \\ -\omega^T & 0 \end{bmatrix} q_t \tag{3.6}$$

$$\dot{q}_t = \frac{1}{2} \Omega(\omega) q_t \tag{3.7}$$

where $[\omega\times]$ is a skew-symmetry matrix that represents a cross product of a vector ω:

$$[\omega\times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.8}$$

Since $\Omega(\omega)$ does not change with time, this becomes an ordinary differential equation with a solution of:

$$q_{t+\Delta t} = \exp\left(\frac{1}{2}\Omega(\omega)\Delta t\right) q_t \tag{3.9}$$

Let $\Theta = \exp\left(\frac{1}{2}\Omega(\omega)\Delta t\right)$ and use of a Taylor series expansion yields:

$$\Theta = I_{4\times4} + \frac{1}{2}\Omega(\omega)\Delta t + \frac{1}{2!}\left(\frac{1}{2}\Omega(\omega)\Delta t\right)^2 + \frac{1}{3!}\left(\frac{1}{2}\Omega(\omega)\Delta t\right)^3 + ... \tag{3.10}$$

$$\Theta = I_{4\times4} + \frac{1}{2}\Omega(\omega)\Delta t + \frac{1}{2!}\left(\frac{1}{2}\Delta t\right)^2 \Omega(\omega)^2 + \frac{1}{3!}\left(\frac{1}{2}\Delta t\right)^3 \Omega(\omega)^3 + ... \tag{3.11}$$

From inspection, we find that:

$$\Theta(\omega)^2 = -\left\|\omega\right\|^2 \cdot I_{4\times4} \qquad \Theta(\omega)^3 = -\left\|\omega\right\|^2 \cdot I_{4\times4}$$

$$\Theta(\omega)^4 = \left\|\omega\right\|^4 \cdot I_{4\times4} \qquad \Theta(\omega)^5 = \left\|\omega\right\|^4 \cdot I_{4\times4}$$

Substituting, expanding and reordering yields:

$$\begin{aligned}\Theta = &\left(1 - \frac{1}{2!}(\frac{1}{2}\left\|\omega\right\|\Delta t)^2 + \frac{1}{4!}(\frac{1}{2}\left\|\omega\right\|\Delta t)^4 - ...\right) I_{4\times4} + \\ &\frac{1}{\left\|\omega\right\|}\left(\frac{1}{2}\left\|\omega\right\|\Delta t - \frac{1}{3!}(\frac{1}{2}\left\|\omega\right\|\Delta t)^3 + \frac{1}{5!}(\frac{1}{2}\left\|\omega\right\|\Delta t)^5 - ...\right)\Omega(\omega)\end{aligned} \tag{3.12}$$

We can recognize that the two infinite series are Taylor series expansions of the cosine and sine functions:

$$\Theta = \cos\left(\frac{1}{2}\left\|\omega\right\|\Delta t\right) I_{4\times4} + \frac{1}{\left\|\omega\right\|}\sin\left(\frac{1}{2}\left\|\omega\right\|\Delta t\right)\Omega(\omega) \tag{3.13}$$

Thus, the new attitude can be propagated from the angular rate input as:

$$q_{t+\Delta t} = \left(\cos\left(\frac{1}{2}\left\|\omega\right\|\Delta t\right) I_{4\times4} + \frac{1}{\left\|\omega\right\|}\sin\left(\frac{1}{2}\left\|\omega\right\|\Delta t\right)\Omega(\omega)\right) q_t \tag{3.14}$$

It is important to note that care must be taken here, as the above equation is numerical unstable for small angular rates ($\|\omega\| \to 0$). We can compute a limit of the above equation in the limit as $\|\omega\| \to 0$ as:

$$q_{t+\Delta t} = \left(I_{4\times 4} + \frac{\Delta t}{2} \Omega(\omega) \right) q_t \tag{3.15}$$

Moreover, the resultant quaternion may need to be normalized to make sure that it satisfies the $q^T q = 1$ constraint.

3.4 Velocity/Position estimation

In addition to attitude estimation, described above, we must estimate the current velocity and position. Acceleration data from the accelerometer can be transformed into the navigation frame and the gravity vector subtracted out. The velocity can be calculated by integrating the resulting acceleration with respect to time. Similarly, position can also be obtained by integrating with the velocity.

3.4.1 Accelerometer model

An accelerometer measures linear acceleration along its sensitive axis. Similar to gyroscopes, measurements are also corrupted with noise, biases and other non-idealities (*Titterton and Weston*, 2004). The measurement model of a 3-axis accelerometer can be written as:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_m = \begin{bmatrix} 1+S_x & M_{xy} & M_{xz} \\ M_{yz} & 1+S_y & M_{yz} \\ M_{zx} & M_{zy} & 1+S_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_t + \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_a + \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}_a \tag{3.16}$$

$$a_m = S \cdot a_t + M \cdot a_t + b_a + n_a \tag{3.17}$$

where a_m is acceleration measured from the sensor while subjecting to true acceleration a_t. The S and M are scale-factor and cross-coupling coefficients, respectively. The sensor's bias b_a and noise n_a are also included in the model.

3.4.2 Velocity integration model

Using acceleration measurements together with current attitude, the current velocity v and position p can be calculated as:

$$\dot{v} = C(\bar{q})(a_m - b_a - n_a) - g_0 \tag{3.18}$$

$$\dot{p} = v \tag{3.19}$$

$$\dot{b_a} = n_{ba} \tag{3.20}$$

the expected value of the above equations are:

$$\dot{\hat{v}} = C(\hat{q})(a_m - \hat{b_a}) - g_0 \tag{3.21}$$

$$\dot{\hat{p}} = \hat{v} \tag{3.22}$$

$$\dot{\hat{b_a}} = 0 \tag{3.23}$$

where $C(\hat{q})$ is the Direction Cosine Matrix (DCM) of the estimated attitude $\hat{q}$ and a_m is the acceleration from the accelerometer and its estimated bias $\hat{b_a}$. We assume that n_a and n_{ba} are zero-mean Gaussian with variance σ_a^2 and σ_{ba}^2, respectively. Gravity vector $g_0 = [0, 0, 9.81]^T [m/s^2]$ is the earth gravity vector pointed vertically downward.

From the above equations (3.18),(3.19),(3.20), we can see that the acceleration has to be integrated twice to calculate position. This integration accumulates noise, resulting in an exponential increase in the position error.

3.5 System model

In this section, we construct a system model using the UKF framework. First, we describe choices of attitude error representation.

3.5.1 Attitude error

Using one redundant parameter, quaternions represent any rotational motion without any singularity or discontinuity. Indeed, some works (*Lefferts et al.*, 1982) use the quaternion directly as the state variables. This approach, however, cannot guarantee that the resultant quaternion satisfies $q^T q = 1$. Moreover, this may lead to singularity in the covariance matrix due to the fact that the states are rank deficient.

To prevent this problem, the quaternion, which has 4 scalars, has to be reduced to 3. There are several methods of transforming a quaternion into a 3-parameter representation of the attitude. Each method utilizes different geometric interpretations and has different singular behavior at different orientations.

In many works, the state representation of attitude is written in the form of attitude error instead of the attitude itself. The attitude error has an advantage in that it represents smaller rotation and can be more easily approximated and linearized.

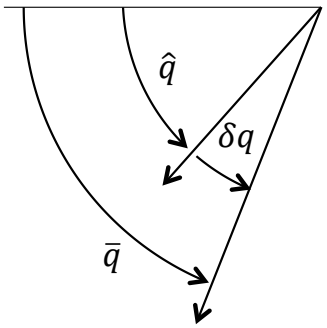

Figure 3.2: Relationship between the true attitude $\bar{q}$, estimated attitude $\hat{q}$ and attitude error δq in a simplified 2d rotation.

From Figure 3.2, we can define the attitude error as:

$$\bar{q} = \delta q \otimes \hat{q} \tag{3.24}$$

$$\delta q = \bar{q} \otimes \hat{q}^{-1} \tag{3.25}$$

where $\otimes$ represents quaternion multiplication.

Above, we can see that the true attitude $\bar{q}$ can be calculated using estimated attitude $\hat{q}$ and estimated attitude error δq. The attitude error is assumed to be small.

There are several ways to incorporate the attitude error as a system state. In the following sections, we discuss small angle approximation and a family of Rodrigues parameters: classical, modified, and generalized Rodrigues parameters.

3.5.1.1 Small angle approximation

In *Lefferts et al.* (1982) and *Trawny and Roumeliotis* (2005), since the error is assumed to be small, the attitude error can be linearized using a small angle approximation as follows:

$$\delta q = \begin{bmatrix} \hat{e} sin(\delta\Phi/2) \\ cos(\delta\Phi/2) \end{bmatrix} \approx \begin{bmatrix} \frac{1}{2}\delta\Phi \\ 1 \end{bmatrix} \tag{3.26}$$

Using this approximation, the attitude error reduces to an error angle vector $\delta\Phi$ of size 3×1. Thus, the state variables for attitude estimation are defined as:

$$\delta p = 2 \cdot [\delta q_0, \delta q_1, \delta q_2]^T \tag{3.27}$$

This method works well in many cases. For large attitude error, however, the estimation may suffer from a non-unit-length quaternion which can lead to divergence of the attitude estimation. This problem is likely to occur when the initial attitude is much different than the actual attitude. To alleviate this problem, the initial attitude

can be initialized using the accelerometer's measurement of the gravity vector.

3.5.1.2 Classical Rodrigues Parameters (CRP)

Another method is to project a quaternion (which falls on a 4 dimensional unit sphere) onto a 3 dimensional hyperplane (*Schaub and Junkins*, 1996; *Markley*, 2003). There are different types of projections depending on the geometry of the projection.

The transformation of quaternion parameters (q) into Classical Rodrigues parameters (p) uses a gnomonic projection—using a projection point at the origin to project parameters onto a hyperplane $q_4 = 1$. The transformation is:

$$p_i = \frac{q_i}{q_4} \qquad i = 1, 2, 3 \tag{3.28}$$

The inverse transformation from the Rodrigues parameters p back to quaternions q must satisfy the $q^T q = 1$ constraint. This transformation is:

$$q_i = \frac{p_i}{\sqrt{1 + p^T p}} \qquad i = 1, 2, 3 \tag{3.29}$$

$$q_4 = \frac{1}{\sqrt{1 + p^T p}} \tag{3.30}$$

Using the definition of the quaternion, the Rodrigues parameters can be expressed directly in term of rotation angle Φ and the rotation axis $\hat{e}$

$$p = \hat{e} \cdot tan\frac{\Phi}{2} \tag{3.31}$$

From above, it is obvious that the classical Rodrigues parameters have a singularity at $\pm 180°$. Moreover, the linearized equation assuming that Φ is small is equal to half the angle of rotation.

$$p \approx \hat{e} \cdot \frac{\Phi}{2} \tag{3.32}$$

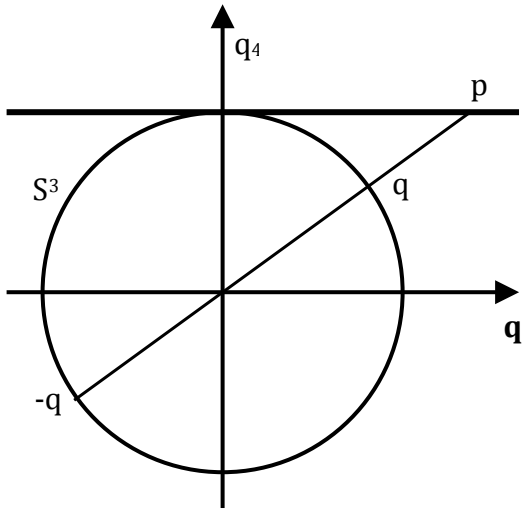

Figure 3.3: Gnomonic projection of classical Rodrigues parameters. A quaternion q can be projected using the origin as a projection point onto a hyperplane $q_4 = 1$. The result is a classical Rodrigues parameter p.

3.5.1.3 Modified Rodrigues Parameters (MRP)

The modified Rodrigues projection (*Schaub and Junkins*, 1996; *Crassidis and Markiey*, 1996; *Markley*, 2003) moves the projection point to the bottom of the unit sphere $q_4 = -1$ and projects parameters onto a hyperplane at $q_4 = 0$:

$$p_i = \frac{q_i}{1 + q_4} \qquad i = 1, 2, 3 \tag{3.33}$$

The inverse transformation is:

$$q_i = \frac{2p_i}{\sqrt{1 + p^T p}} \qquad i = 1, 2, 3 \tag{3.34}$$

$$q_4 = \frac{1 - p^T p}{\sqrt{1 + p^T p}} \tag{3.35}$$

Similarly, the Rodrigues parameters can be expressed directly in terms of rotation angle Φ and the rotation axis $\hat{e}$:

$$p = \hat{e} \cdot \tan \frac{\Phi}{4} \tag{3.36}$$

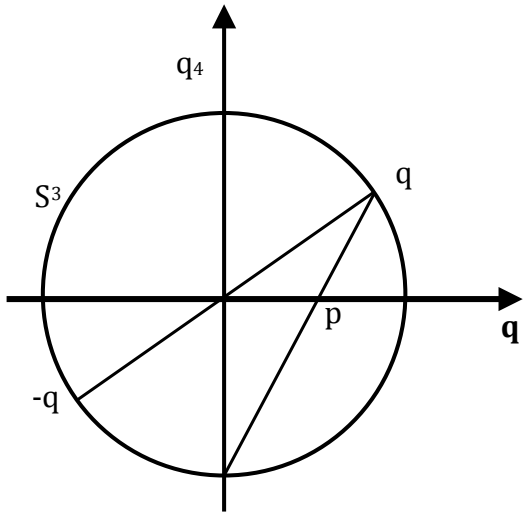

Figure 3.4: Projection of modified Rodrigues parameters. A quaternion q can be projected using a projection point $q_4 = -1$ onto a hyperplane $q_4 = 0$. The result is a modified Rodrigues parameter p.

The transformation has singularities at $\Phi = \pm 360°$, twice the range of the classical Rodrigues parameters. Moreover, the limits $\Phi = \pm 360°$ can be wrapped to $\Phi = 0$ which does not have singularity.

3.5.1.4 Generalized Rodrigues Parameters (GRP)

In *Crassidis and Markley* (2003), the generalized Rodrigues parameter is further generalized using a projection point $q_4 = -a$, where $a \in [0, 1]$, and scaling of the projected parameters f. The transformation can be described as:

$$p_i = f \cdot \frac{q_i}{a + q_4} \qquad i = 1, 2, 3 \tag{3.37}$$

We can see that classical ($a = 0$ and $f = 1$) and modified Rodrigues parameter ($a = 1$ and $f = 1$) are special cases of the Generalized Rodrigues Parameters. In *Crassidis and Markley* (2003), the parameter f is set to $f = 2(a + 1)$ and $a = 1$, the

linearized transformation of a small rotation is:

$$p = 2\hat{e}(a+1)\frac{sin(\frac{\Phi}{2})}{a+cos(\frac{\Phi}{2})} \tag{3.38}$$

$$p \approx \hat{e} \cdot \Phi \tag{3.39}$$

The inverse transformation from p back to q is given by:

$$q_4 = \frac{-a \cdot p^T p + f\sqrt{f^2 + (1-a^2)p^T p}}{f^2 + p^T p} \tag{3.40}$$

$$q_i = f^{-1}(a+q_4)p_i \qquad i = 1,2,3 \tag{3.41}$$

This generalized Rodrigues Parameters have the same advantages as the MRP with an option to be able to control the scale of the state variables.

In this project, we will compare performances of each representation. To this end, we use UKF implementation of the generalized Rodrigues Parameters model, which can be configured to be equivalent to MRP or CRP according to parameter a and f.

3.5.2 State variables

We define state variables of the system as:

$$x = \begin{bmatrix} \delta p \\ b_g \\ b_a \\ v \\ p \end{bmatrix} \begin{array}{c} \text{projected attitude error} \\ \text{gyroscope biases} \\ \text{accelerometer biases} \\ \text{velocity} \\ \text{position} \end{array} \tag{3.42}$$

3.5.3 Non-additive noise model

The system model with non-additive noise can be written as:

$$x_k = f(x_{k-1}, u_{k-1}, w_k) \tag{3.43}$$

$$\tilde{y}_k = h(x_k, v_k) \tag{3.44}$$

From above system model, we can see that both process noise w_k and measurement noise v_k are directly integrated into the non-linear time update equation $f(x_{k-1}, u_{k-1}, w_k)$ and measurement equation $h(x_k, v_k)$.

As described in Algorithm 2.2, a time-update model is shown in equation (2.29) which has sigma points input $X^a_{k-1}(i)$, $i = 0, ..., 2n$ and produces state prediction at the next time-step $X_k(i)^{x-}$.

$$X_{k+1}(i)^{x-} = F(X^x_k(i), u_k, X^Q_k(i))$$

where $u_k = [\omega_{m,k}, a_{m,k}]^T$ are gyroscope and accelerometer inputs.

$X^Q_k(i) = [n_\omega(i), n_{bg}(i), n_{ba}(i), n_a(i)]^T$ is a sigma point of the process noise which has variance of $\sigma^2_{\omega,d}, \sigma^2_{bg,d}, \sigma^2_{ba,d}, \sigma^2_{a,d}$, respectively (section 3.3.1 and 3.4.2). In this method, we need to use a discrete version of noise which incorporates sampling time.

$$\sigma^2_{\omega,d} = \frac{\sigma^2_{\omega,c}}{\Delta t} \tag{3.45}$$

$$\sigma^2_{bg,d} = \sigma^2_{bg,c}\Delta t \tag{3.46}$$

$$\sigma^2_{ba,d} = \sigma^2_{ba,c}\Delta t \tag{3.47}$$

$$\sigma^2_{a,d} = \frac{\sigma^2_{a,c}}{\Delta t} \tag{3.48}$$

By adapting from the procedure in *Crassidis and Markley* (2003), the time-update model can be described in the following steps:

1. Calculate attitude error of the i^{th} sigma point,

   $\delta q(i) = [\delta q_1(i), \delta q_2(i), \delta q_3(i), \delta q_4(i)]^T$ from $\delta p(i)$ using equation (3.40) and (3.41):

$$\delta q_4(i) = \frac{-a \cdot \delta p(i)^T \delta p(i) + f\sqrt{f^2 + (1-a^2)\delta p(i)^T \delta p(i)}}{f^2 + \delta p(i)^T \delta p(i)} \tag{3.49}$$

$$\delta q_i(i) = f^{-1}(a + \delta q_4(i))\delta p(i)_i \qquad i = 1, 2, 3 \tag{3.50}$$

2. Using current attitude estimate $\hat{q}_k$, calculate the attitude estimate of each sigma point using equation (3.24):

$$\hat{q}_k(i) = \delta q(i) \otimes \hat{q}_k \tag{3.51}$$

3. Propagate the attitude estimate of each sigma point in the above to the next time-step using equation (3.14):

$$\hat{\omega}_k(i) = \omega_m - b_g(i) - n_\omega(i) \tag{3.52}$$

$$\hat{q}_{k+1}(i) = \left(\cos\left(\frac{1}{2}\left\|\hat{\omega}_k(i)\right\|\Delta t\right) I + \frac{1}{\left\|\hat{\omega}_k(i)\right\|}\sin\left(\frac{1}{2}\left\|\hat{\omega}_k(i)\right\|\Delta t\right)\Omega(\hat{\omega}_k(i))\right)\hat{q}_k(i) \tag{3.53}$$

$$\hat{q}_{k+1} = \hat{q}_{k+1}(0) \tag{3.54}$$

4. Project the attitude estimates back to state variables $\delta p(i)$ using equation (3.37):

$$\delta p_{k+1}(0) = 0 \tag{3.55}$$

$$\delta p_{j,k+1}(i) = f \cdot \frac{\hat{q}_{j,k+1}}{a + \hat{q}_{4,k+1}} \qquad j = 1, 2, 3 \tag{3.56}$$

   In summary, the attitude propagation and projection can be described as in Figure 3.5

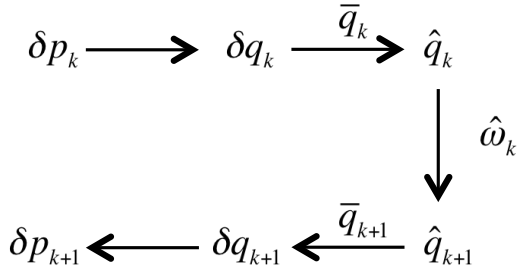

Figure 3.5: Attitude state propagation steps. δp is projected attitude error, δq is quaternion attitude error and $\hat{q}$ is estimated attitude.

5. Finally, since the attitude has been propagated, we can propagate other state variables to the next time-step:

$$b_{g,k+1}(i) = b_{g,k}(i) + n_{bg}(i) \tag{3.57}$$

$$b_{a,k+1}(i) = b_{a,k}(i) + n_{ba}(i) \tag{3.58}$$

$$v_{k+1}(i) = v_k(i) + (C(\hat{q}_k(i))(a_m - b_{a,k}(i)) - g_0)\,\Delta t \tag{3.59}$$

$$p_{k+1}(i) = p_k(i) + v_k(i)\Delta t \tag{3.60}$$

The UKF non-additive model has the advantage of incorporating noise directly into the system model and avoiding linearization. This does, however, significantly increase the number of required sigma points. For a system with n states, m process noise inputs and o measurement noise inputs, the total number of required sigma points is $2(n + m + o) + 1$, compared to $2n + 1$ in the case of the additive noise model. Since the process of generating sigma points has computational complexity of $O(n^3)$ (Section 2.2), the larger number of sigma points will require substantially more computational time.

3.5.4 Additive noise model

To reduce the number of sigma points, we can linearize the noise model out of the system model and assume it to be additive:

$$x_k = f(x_{k-1}, u_{k-1}) + G_k w_k \tag{3.61}$$

$$\tilde{y}_k = h(x_k) + v_k \tag{3.62}$$

where w_k and v_k are zero-mean Gaussian noise with covariance Q_k and R_k, respectively.

For state propagation, the procedure of modeling the noise as additive is very similar to the non-additive version; however, the additive noise model does not consider noise inputs in the propagation step since it is processed separately.

Our focus now turns to the process noise covariance Q_k, which is a linearized version of the noise in the system model. To find Q_k, we need the linearized system equation.

3.5.4.1 Linearized state equation

In this section, we will construct a state equation for attitude estimation as described in the previous sections. This linearized state equation will be used to derive process noise covariance Q.

Using equation (3.3) and (3.4) and (3.7), the continuous-time state equation can be written as:

$$\dot{\bar{q}} = \frac{1}{2}\Omega(\omega_m - b_g - n_\omega)\bar{q} \tag{3.63}$$

$$\dot{b}_g = n_{bg} \tag{3.64}$$

The expected value of the state equations are:

$$\dot{\hat{q}} = \frac{1}{2}\Omega(\omega_m - \hat{b_g})\hat{q} \tag{3.65}$$

$$= \frac{1}{2}\Omega(\hat{\omega})\hat{q} \tag{3.66}$$

$$\dot{\hat{b}}_g = 0 \tag{3.67}$$

As described in Section 3.5.1, we need to rewrite the equation as a function of attitude error δq instead of the actual attitude q. Using equation (3.24), one obtains:

$$\begin{aligned} \bar{q} &= \delta q \otimes \hat{q} \\ \dot{\bar{q}} &= \delta\dot{q} \otimes \hat{q} + \delta q \otimes \dot{\hat{q}} \end{aligned} \tag{3.68}$$

using equation (3.6), (3.63) and (3.65), solving for $\delta\dot{q}$ yields:

$$\delta\dot{q} = \begin{bmatrix} -[\hat{\omega}\times]\delta q - \frac{1}{2}(b_g + n_\omega) \\ 0 \end{bmatrix} \tag{3.69}$$

As our state variable for attitude is δp, we need to project the 4 dimensional quaternion error δq to 3 parameters variable δp. By using equation (3.39), δp is approximated to be:

$$\delta p \approx 2\left[\delta q_1, \delta q_2, \delta q_3\right]^T \tag{3.70}$$

Thus, the linearized equation for the attitude becomes:

$$\delta\dot{p} = -[\hat{\omega}\times]\delta p - b_g - n_\omega \tag{3.71}$$

Likewise, the state equation for velocity is described in equation (3.18) and (3.21):

$$\dot{v} = C(\bar{q})(a_m - b_a - n_a) - g_0 \tag{3.72}$$

$$\dot{\hat{v}} = C(\hat{q})(a_m - \hat{b_a}) - g_0 \tag{3.73}$$

For small attitude error, its DCM can be approximated as:

$$C(\delta q) \approx I - [\delta p \times] \tag{3.74}$$

After significant algebraic operations, the acceleration error $\delta\dot{v} = \dot{v} - \dot{\hat{v}}$ can be written as:

$$\delta\dot{v} = [C(\hat{q})(a_m - b_a)\times]\delta p - C(\hat{q})b_a - [I - [\delta p\times]]\, C(\hat{q})n_a \tag{3.75}$$

$$\delta\dot{v} \approx [C(\hat{q})(a_m - b_a)\times]\delta p - C(\hat{q})b_a - C(\hat{q})n_a \tag{3.76}$$

Thus, the linear continuous-time error state equation becomes:

$$\delta\dot{x} = F \cdot \delta x + G \cdot w \tag{3.77}$$

$$\begin{bmatrix} \delta\dot{p} \\ \delta\dot{b_g} \\ \delta\dot{b_a} \\ \delta\dot{v} \\ \delta\dot{p} \end{bmatrix} = \begin{bmatrix} -[\hat{\omega}\times] & -I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ [C(\hat{q}(a_m - b_a)\times] & 0 & -C(\hat{q}) & 0 & 0 \\ 0 & 0 & 0 & I & 0 \end{bmatrix} \begin{bmatrix} \delta p \\ \delta b_g \\ \delta b_a \\ \delta v \\ \delta p \end{bmatrix} + \begin{bmatrix} -I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & -C(\hat{q}) \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} n_\omega \\ n_{bg} \\ n_{ba} \\ n_a \end{bmatrix} \tag{3.78}$$

Using a Taylor series expansion, the approximate discrete time state transition matrix F can be obtained as:

$$F_d \approx I + F\Delta t + \frac{1}{2!}F^2\Delta t^2 + ... \tag{3.79}$$

$$F_d \approx \begin{bmatrix} I - \Delta t[\hat{\omega}\times] & -\Delta t & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ \Delta t[C(\hat{q}(a_m - b_a)\times] & 0 & -\Delta t C(\hat{q}) & I & 0 \\ 0 & 0 & 0 & \Delta t & I \end{bmatrix} \tag{3.80}$$

3.5.4.2 Process noise covariance

The discrete process noise covariance matrix can be calculated using:

$$Q_d = \int_{t_k}^{t_{k+1}} F_d \cdot G \cdot Q_c \cdot G^T \cdot F_d^T d\tau \tag{3.81}$$

where

$$Q_c = \begin{bmatrix} \sigma_{\omega,c}^2 I & 0 & 0 & 0 \\ 0 & \sigma_{bg,c}^2 I & 0 & 0 \\ 0 & 0 & \sigma_{ba,c}^2 I & 0 \\ 0 & 0 & 0 & \sigma_{a,c}^2 I \end{bmatrix} \tag{3.82}$$

The result Q_d is shown in equation (3.84). In our realtime implementation of the algorithm, we split up the Q_d matrix into a constant part and time-varying part. Using this technique, only a few elements of the matrix are updated at every time-step, therefore speeding up the calculation.

Defining, $SF = [C(\hat{q})(a_m - b_a)\times]$,

$$Q_d = \int_{t_k}^{t_{k+1}} \begin{bmatrix} (\sigma_{\omega,c}^2 + \sigma_{bg,c}^2\tau^2)I & (-\sigma_{bg,c}^2\tau)I & 0 & \tau\sigma_{\omega,c}^2 SF^T & 0 \\ (-\sigma_{bg,c}^2\tau)I & b_{bg,c}I & 0 & 0 & 0 \\ 0 & 0 & b_{ba,c}I & -C(\hat{q})^T\tau\sigma_{ba,c}^2 & 0 \\ \tau\sigma_{\omega,c}^2 SF & 0 & -C(\hat{q})\tau\sigma_{ba,c}^2 & (\sigma_{a,c}^2 + \sigma_{ba,c}^2\tau^2)I + \sigma_{\omega,c}^2\tau^2 SF \cdot SF^T & (\sigma_{a,c}^2\tau)I \\ 0 & 0 & 0 & (\sigma_{a,c}^2\tau)I & (\sigma_{a,c}^2\tau^2)I \end{bmatrix} d\tau \tag{3.83}$$

$$Q_d = \begin{bmatrix} (\sigma_{\omega,c}^2\Delta t + \sigma_{bg,c}^2\frac{\Delta t^3}{3})I & (-\sigma_{bg,c}^2\frac{\Delta t^2}{2})I & 0 & \frac{\Delta t^2}{2}\sigma_{\omega,c}^2 SF^T & 0 \\ (-\sigma_{bg,c}^2\frac{\Delta t^2}{2})I & b_{bg,c}\Delta t I & 0 & 0 & 0 \\ 0 & 0 & b_{ba,c}\Delta t I & -C(\hat{q})^T\frac{\Delta t^2}{2}\sigma_{ba,c}^2 & 0 \\ \frac{\Delta t^2}{2}\sigma_{\omega,c}^2 SF & 0 & -C(\hat{q})\frac{\Delta t^2}{2}\sigma_{ba,c}^2 & (\sigma_{a,c}^2\Delta t + \sigma_{ba,c}^2\frac{\Delta t^3}{3})I + \sigma_{\omega,c}^2\frac{\Delta t^3}{3} SF \cdot SF^T & (\sigma_{a,c}^2\frac{\Delta t^2}{2})I \\ 0 & 0 & 0 & (\sigma_{a,c}^2\frac{\Delta t^2}{2})I & (\sigma_{a,c}^2\frac{\Delta t^3}{3})I \end{bmatrix} \tag{3.84}$$

3.6 Sensor measurement model

In order to provide additional information to the system, we can directly implement additional sensors or even exploit indirect information from the nature of the human walking cycle. Each measurement brings different information to the system at a different phase of walking and for different operating conditions. The following sections are a non-exhaustive list of measurement types that we use in the PDR system.

3.6.1 Zero Velocity Update (ZUPT)

Ayyappa (1997) described the human gait cycle as consisting of "stance" and "swing" phases. According to the study, the majority of the gait cycle is stance phase, which is 62% of the total gait time, while the swing phase makes up the remaining 38%.

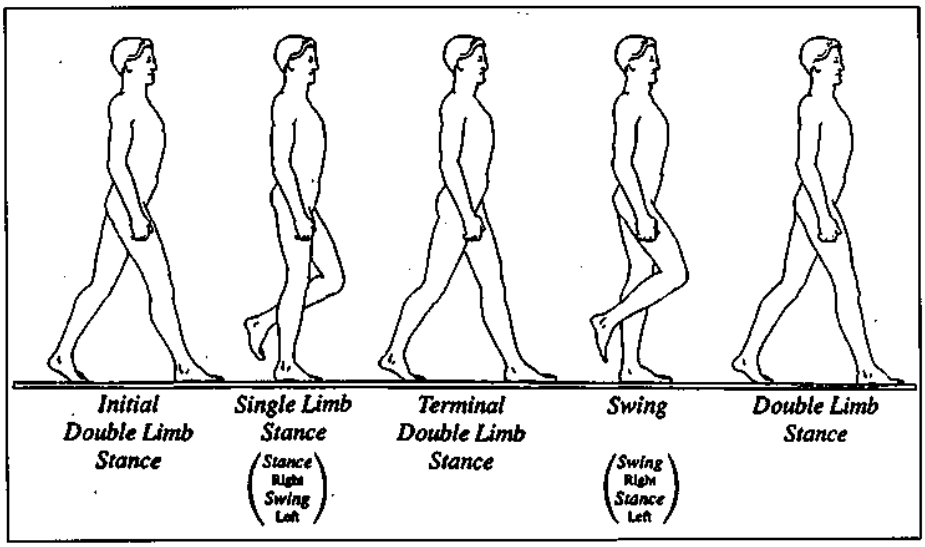

Figure 3.6: Human gait cycle (*Ayyappa*, 1997)

Ojeda and Borenstein (2007a) exploit the fact that, at the stance phase, the relative velocity between the foot and ground is zero unless the sole is slipping on the ground. This way, we can safely assume (to some extent) that the velocity of the

sensor during the so-called "footfall" is zero. This method of compensating for drift is called Zero Velocity Update (ZUPT):

$$v = n_v \tag{3.85}$$

where n_v is a zero-mean Gaussian noise representing the uncertainty that the velocity is actually zero. In some implementations, such as in *Foxlin* (2005), the variance of n_v is equal to the trace of the velocity covariance sub-matrix. For our implementation, we set the variance of n_v to be the maximum of the diagonal component of the velocity covariance sub-matrix. These modifications are designed to gradually make corrections with increasing confidence, which would prevent measurement overconfidence in the case of false step-detection.

There are several methods to detect the footfall duration and apply ZUPT. One way is to distinguish stance period from the raw measurement using heuristics (*Ojeda and Borenstein*, 2007a; *Jimenez et al.*, 2009). Another uses a neural network (*Beauregard and Haas*, 2006) to learn parameters from a training set.

To achieve the highest performance using the ZUPT technique, there are still some challenges in implementing a robust step detection approach that produces fewer false-positive detections and works with a variety of walking styles on all subjects.

3.6.2 Zero Attitude Rate Update (ZARU)

Some research such as *Foxlin* (2005) and *Beauregard* (2009) describe a method to assume zero attitude rate during stand still. This can be an explicit instruction for the subject to stand still during the system start-up or else an implicit detection during the still period later applied as an update in the background (*Ojeda and Borenstein*,

2007a). From equation (3.3), the true rate is assumed to be zero $\omega_t = 0$:

$$\omega_m = b_g + n_\omega \tag{3.86}$$

In our experience, we found that this method is not suitable to be applied at every footfall. Though the foot has zero-velocity during footfall, it is still rolling from the fact that the heel is rising up in preparing the foot to swing forward. Therefore, there is, in fact, a small rotation during the stance period – rendering the zero rotation assumption invalid.

3.6.3 Magnetometer assisted

For outdoor applications, an additional sensor can be added to measure magnetic heading. In Chapter IV, we augment the PDR system with a 3-axis magnetometer to measure magnetic heading. The magnetic heading measurement can be written as a function of the yaw angle of the current attitude $yaw(\bar{q})$ and noise in magnetic heading measurement n_Ψ.

$$\Psi_{mag} = \mathrm{yaw}(\bar{\mathrm{q}}) + \mathrm{n}_\Psi \tag{3.87}$$

By augmenting the system with absolute heading angle measurement, the gyro drift in the Z direction can be eliminated. This improves heading error substantially over the original PDR system. In Chapter IV, we conclude that this method can reduce the position error to less than 1.9% of distance travelled using a Memsense nIMU.

3.6.4 Artificial constraints

In many applications where the operating area is in a structured environment, we can feed artificial constraints to the filter. For example, we can assume the floors in a

given building to be flat. In this scenario, if the change in elevation between two steps is smaller than a threshold, we can make a observation that the elevation change is zero. This method has to be used with care to make sure that the assumptions hold true for the actual environment.

3.7 Parameterization of the sensor model

In this section, we describe methods of parameterizing sensor models for two types of IMU: Navchip and an older Memsense nIMU. The results of this section also show a performance comparison between the two IMU's technologies. The first part is to determine the sensitivity to the linear acceleration matrix which is described in the gyroscope model in Section 3.3.1.

3.7.1 Sensitivity to linear acceleration of gyroscopes

To begin, each IMU is mounted on a programable two degree-of-freedom tilt table. By varying roll and pitch angle, the IMU is subjected to various magnitudes of acceleration. We can then take a measurement of angular rates from the gyroscope to find the sensitivity to linear acceleration of each axis of the gyroscope.

From equation (3.2) , the rate measurement of a stationary 3-axis gyroscope can be written as:

$$\omega_m = \hat{G} \cdot a_m + \hat{b_g} \tag{3.88}$$

The rate ω_m and acceleration a_m can be measured directly from the sensor. Consequently, the sensitivity to linear acceleration matrix $\hat{G}$ and gyro biases b_g can be estimated using linear regression.

The examples of the sensitivity to linear acceleration (deg/sec per m/s^2) of two types of IMU were found to be:

Navchip:

$$G = \begin{bmatrix} -0.00042816 & -0.00037100 & 0.00040087 \\ 0.00049081 & -0.00053108 & -7.12798797e^{-06} \\ -2.22648666e^{-05} & -0.00068323 & -0.00013779 \end{bmatrix}$$

Memsense nIMU:

$$G = \begin{bmatrix} 0.00120899 & 0.01491520 & -0.00206642 \\ -0.01444794 & 0.00500313 & 0.00215838 \\ 0.00433347 & -0.01415647 & 0.01042075 \end{bmatrix}$$

From the above numbers it is possible to conclude that, overall, the Navchip IMU is generally less sensitive to acceleration than the older Memsense nIMU sensor.

3.7.2 Sensor noise model

As with all measurements, the output of the gyroscope and accelerometer in the PDR system can be corrupted by noise and bias drift. To characterize this, we utilized a standard methodology called "Allan variance" or "Allan deviation" which was first described in *Allan and Leschiutta* (1974).

A more specific application of the Allan variance methodology to the gyroscope and accelerometer can be found in *El-Sheimy et al.* (2008); *Petkov and Slavov* (2010). For completeness, we show here the definition of this method.

Assume that we have N consecutive samples $\Omega(t)$, and divide them into n groups

of consecutive samples. The Allan variance $\sigma^2(T)$ of length T is defined as:

$$\bar{\Omega}_k(T) = \frac{1}{T} \int_{t_k}^{t_k+T} \Omega(t)dt \tag{3.89}$$

$$\bar{\Omega}_{next}(T) = \frac{1}{T} \int_{t_{k+1}}^{t_{k+1}+T} \Omega(t)dt \tag{3.90}$$

$$\sigma^2(T) = \frac{1}{2(N-2n)} \sum_{k=1}^{N-2n} \left[\bar{\Omega}_{next}(T) - \bar{\Omega}_k(T)\right]^2 \tag{3.91}$$

The novelty of the Allan variance is that when plotted against length T, it decomposes different types of noise into different regions of the log-log plot.

1. Angle/Velocity random walk

   Angle/Velocity is an angular/velocity error process which is due to white noise in angular rate/acceleration. Angle/Velocity random walk is where the slope is equal to $-1/2$. The numerical value of noise coefficient can be directly read where the graph crosses $T = 1$.

2. Bias instability

   The bias instability indicates how stable the bias is over the period of time. The value of the bias instability coefficient can be found at the minimum of the constant part of the graph.

Figure 3.7 shows Allan variance plots of all gyroscopes inside the Navchip and Memsense IMUs. The angle random walk and bias instability coefficients are summarized in Table 3.1.

Similarly, Figure 3.8 shows Allan variance of all accelerometers and Table 3.2 shows the numerical value of the coefficients.

We found that the Navchip has an order of magnitude lower random walk and bias stability than the Memsense nIMU.

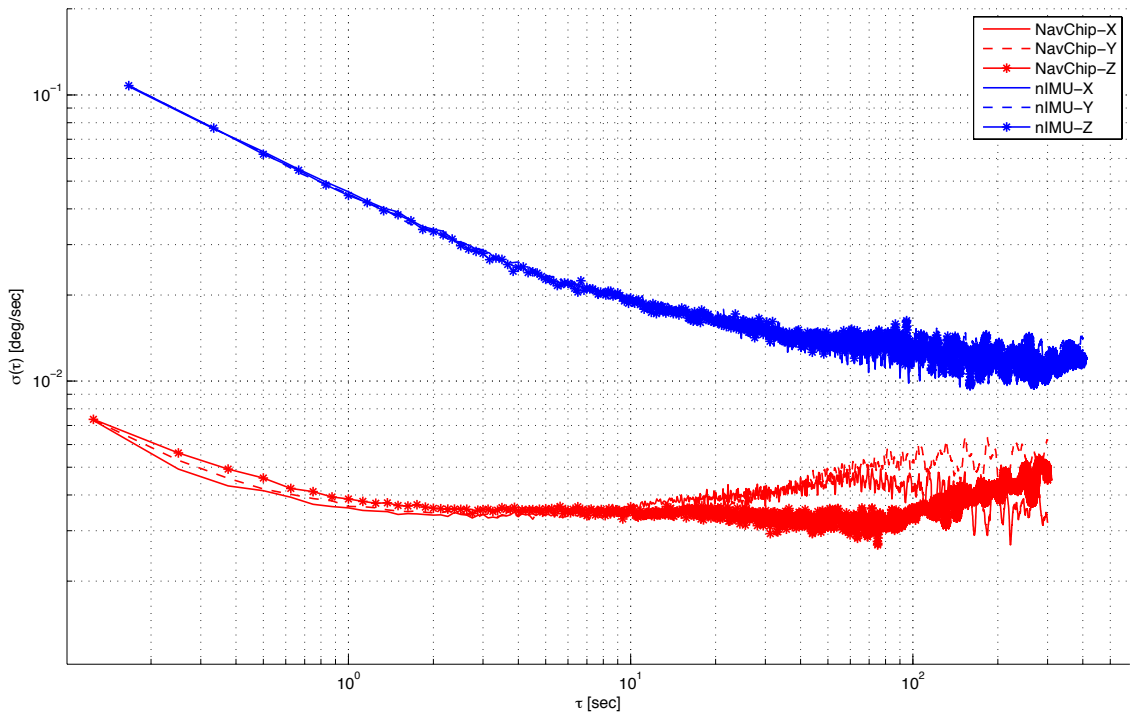

Figure 3.7: Allan variance plot of two gyroscopes: Navchip (red) and Memsense nIMU (blue). The Navchip has lower noise than the older Memsense nIMU.

		Angle random walk ($^\circ/s/\sqrt{s}$)	Bias instability ($^\circ/s$)
Memsense nIMU	X	0.0458	0.0140 @38 sec
	Y	0.0452	0.0143 @38 sec
	Z	0.0446	0.0142 @38 sec
Navchip	X	0.0036	0.0034 @2 sec
	Y	0.0037	0.0035 @2 sec
	Z	0.0039	0.0036 @2 sec

Table 3.1: Angle random walk and bias instability of Navchip and Memsense nIMU's gyroscopes.

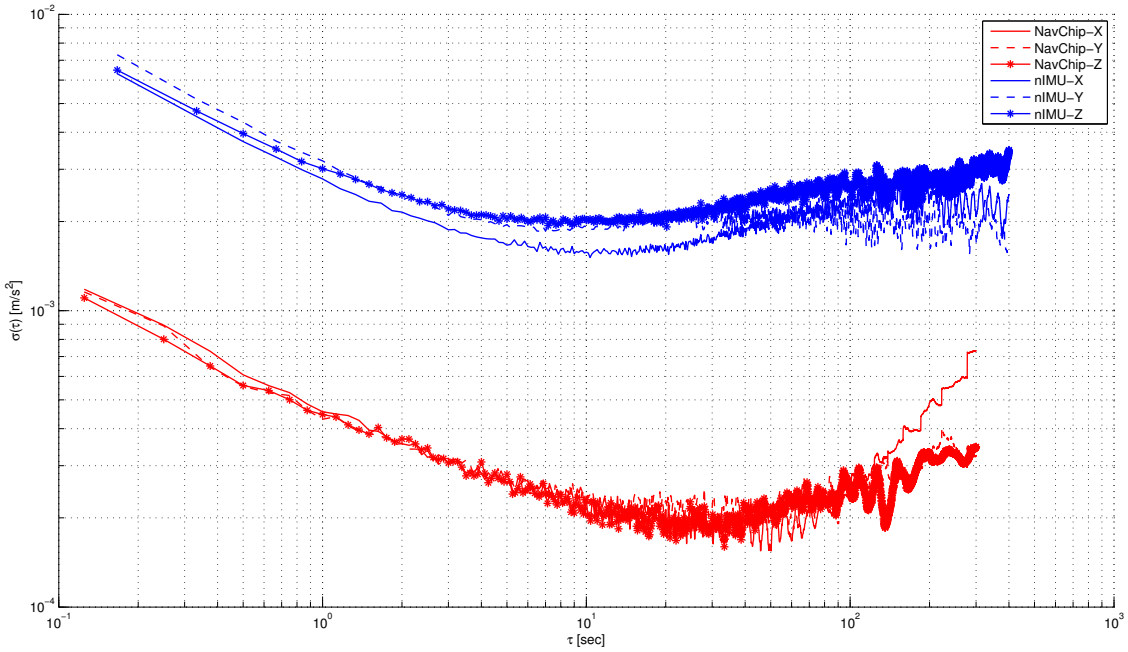

Figure 3.8: Allan variance plot of two accelerometers: Navchip (red) and Memsense nIMU (blue). The Navchip has lower noise compared to the Memsense nIMU.

		Velocity random walk $(m/s^2/\sqrt{s})$	Bias instability $(^\circ/s)$
Memsense nIMU	X	0.00278	0.00157 @ 9 sec
	Y	0.00320	0.00186 @ 8 sec
	Z	0.00301	0.00197 @ 8 sec
Navchip	X	0.00046	0.00019 @22 sec
	Y	0.00043	0.00021 @22 sec
	Z	0.00045	0.00019 @22 sec

Table 3.2: Velocity random walk and bias instability of Navchip and Memsense nIMU's accelerometers.

3.8 Experimental results

In this section, we evaluate the performance of the UKF implementation by testing them against a baseline original and a widely used EKF algorithm. We use our PDR dataset which consists of 130 experiments conducted by several subjects walking and crawling inside buildings. The dataset contains several experiments of a subject walking in indoor environments. Some of the experiments include both walking and crawling, making these datasets useful for evaluating performance of the PDR system in a variety of settings.

3.8.1 UKF, EKF and original PDR system

Since the original PDR system does not consider any sensor biases, we define state variables of the system as a 9-state version in this test, as follows:

$$x = \begin{bmatrix} \delta p \\ v \\ p \end{bmatrix} \begin{matrix} \text{projected attitude error} \\ \text{velocity} \\ \text{position} \end{matrix} \tag{3.92}$$

The average of *position error per distance travelled* from 130 experiments is shown in the Table (3.3). We can see that all three algorithms perform comparably (within confidence interval) for horizontal error. On the other hand, for elevation error, EKF and UKF perform significantly better than the original system. The performance of the UKF and EKF are similar in elevation performance.

	Elevation error per distance travelled (%)		Horizontal error per distance travelled (%)	
	mean	95% confidence interval	mean	95% confidence interval
Original	3.964	±0.6879	2.551	±1.198
EKF9	2.367	±0.4603	2.430	±1.162
UKF9	2.289	±0.4067	3.281	±1.287

Table 3.3: Performance of original system versus EKF and UKF implementation.

3.8.2 Model complexity

We now consider the effect of model complexity on performance. We define 9, 12 and 15 state versions of the system model with increasing complexity, as follows:

$$x_9 = \begin{bmatrix} \delta p \\ - \\ - \\ v \\ p \end{bmatrix}, x_{12} = \begin{bmatrix} \delta p \\ b_g \\ - \\ v \\ p \end{bmatrix}, x_{15} = \begin{bmatrix} \delta p \\ b_g \\ b_a \\ v \\ p \end{bmatrix} \quad \begin{matrix} \text{projected attitude error} \\ \text{gyroscope biases} \\ \text{accelerometer biases} \\ \text{velocity} \\ \text{position} \end{matrix} \tag{3.93}$$

Table 3.4 shows the performance of each model. For elevation error, both UKF12 and UKF15 perform best considering uncertainty; however, UKF12 has larger horizontal error in comparision to UKF15. Specifically, the UKF15 model has an average elevation error 63% less than the original system. An example of the position output of original system and UKF15 is shown in Figure 3.12.

	Elevation error per distance travelled (%)		Horizontal error per distance travelled (%)	
	mean	95% confidence interval	mean	95% confidence interval
UKF9	2.289	±0.4067	3.281	±1.287
UKF12	1.522	±0.2172	4.400	±1.028
UKF15	1.448	±0.2874	2.558	±1.121

Table 3.4: Performance of UKF implementation with various model complexity.

3.8.3 Additive vs. Non-additive model

We also test the UKF 15-state version with additive (UKF15) and non-additive (UKF15NA) noise model as described in Sections 3.5.3 and 3.5.4.

As shown in Table 3.5, both additive and non-additive noise model perform virtually identically. Thus, we are able to conclude that the linearization of the noise model does not affect the performance of the system.

	Elevation error per distance travelled (%)		Horizontal error per distance travelled (%)	
	mean	95% confidence interval	mean	95% confidence interval
UKF15	1.448	±0.2874	2.558	±1.121
UKF15NA	1.440	±0.2867	2.548	±1.121

Table 3.5: Performance of UKF implementation with additive and non-additive noise model.

3.8.4 Flat-floor assumption

As described in Section 3.6.4, we assumed that all the floors our subjects walked on were flat. This assumption exploited as follows: if the change in elevation from one step to the next is less than a threshold (also within the uncertainty), a dummy measurement of elevation is created and set equal to the elevation of the previous step.

	Elevation error per distance travelled (%)		Horizontal error per distance travelled (%)	
	mean	95% confidence interval	mean	95% confidence interval
UKF15	1.448	±0.2874	2.558	±1.121
UKF15FF	0.906	±0.2286	2.440	±1.112

Table 3.6: Performance of UKF implementation with flat-floor assumption.

Table 3.6 clearly indicates that the UKF15FF has the lowest rate of elevation error, though there is also no improvement in horizontal error.This flat-floor assumption is not always valid; this experiment is only intended to show what performance gains are possible.

In summary, a comparison of elevation and horizontal error between all implementations are shown in Figure 3.9 and Figure 3.10.

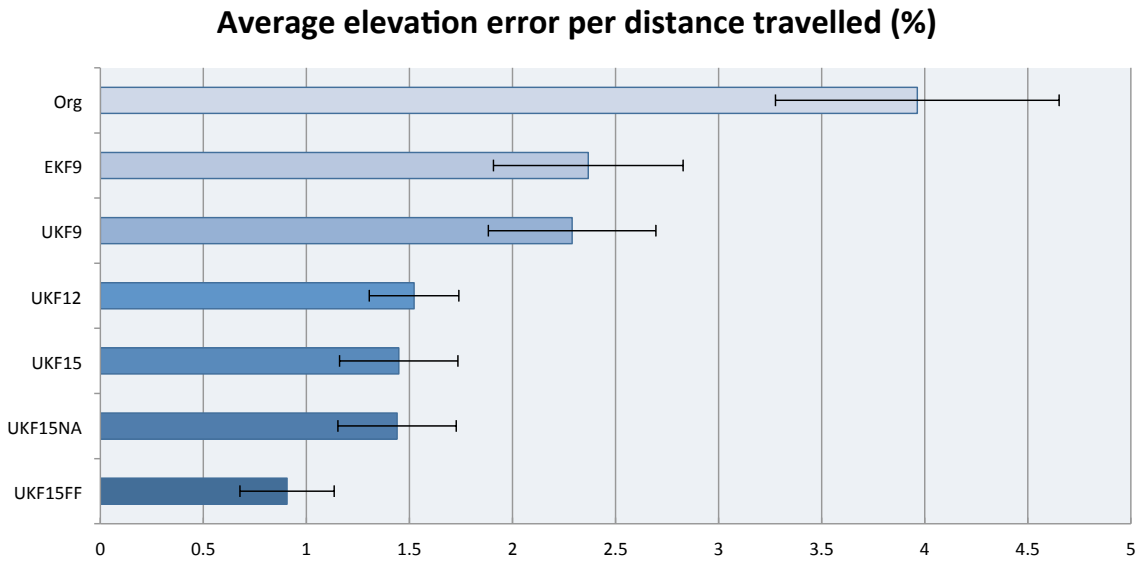

Figure 3.9: Average elevation error per distance travelled of various implementations.

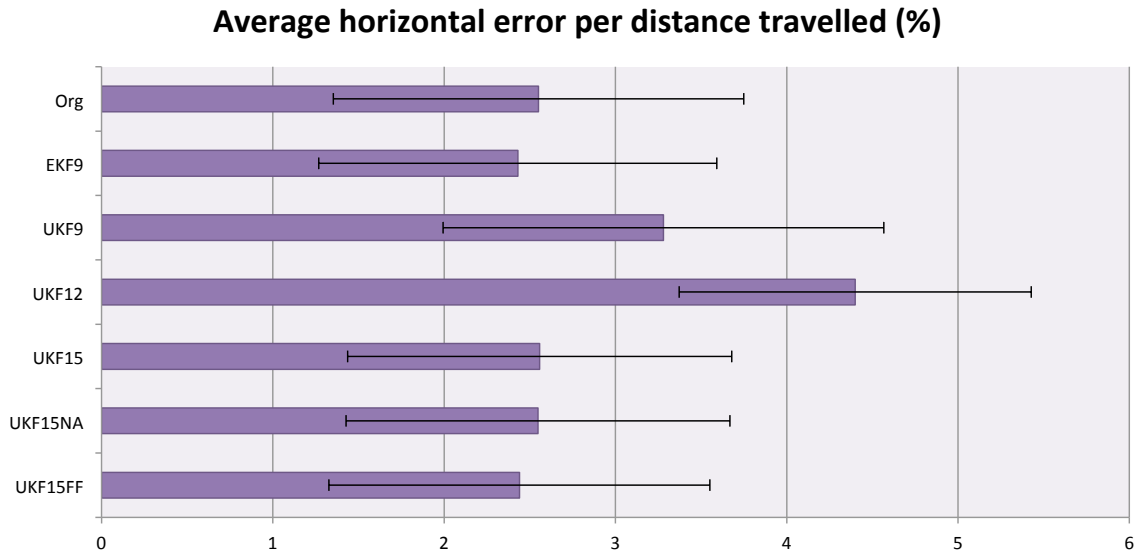

Figure 3.10: Average horizontal error per distance travelled of various implementations.

3.8.5 Runtime comparison

As the complexity of the models increased, the computational complexity also increased as well. In this experiment, we compared the computational time of each

model with the original algorithm as the baseline.

	Computational time (times of original system)	
	mean	95% confidence interval
Original	1	-
EKF9	1.359	0.008361
UKF9	1.835	0.01641
UKF12	2.027	0.01576
UKF15	2.194	0.01429
UKF15NA	3.478	0.02806
UKF15FF	2.235	0.01710

Table 3.7: Relative runtime of various implementations comparing to original system.

From Table 3.7, we can see that the required runtime increases with model complexity. For the same number of states, the UKF is slower than EKF. Moreover, the non-additive noise model (UKF15NA) is much slower than additive noise (UKF15).

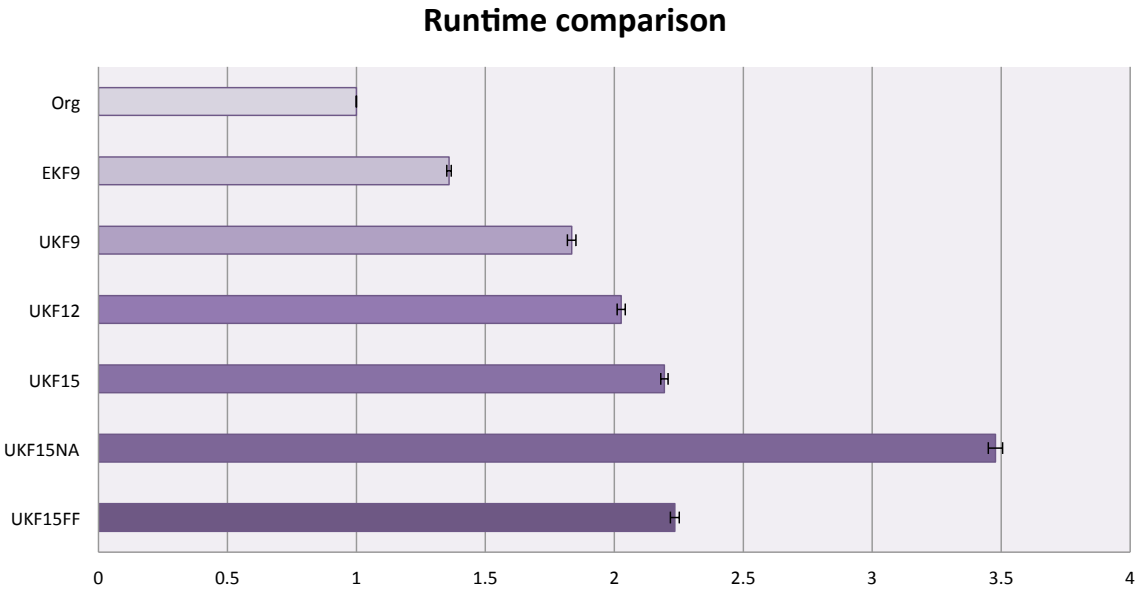

Figure 3.11: Relative runtime of various implementations comparing to original system.

3.9 Conclusion

In this chapter, we evaluated the performance limitations of the current PDR system. Without any correction from other sensors, this system accumulates large elevation errors, which we attribute to attitude error. To address this, we proposed an upgrade to the estimation algorithm, using the Unscented Kalman Filter (UKF). We described our gyroscope model, accelerometer model, velocity/position model and deriving the necessary propagation steps. Finally, we parameterized two IMUs and compared their characteristics.

We also conducted tests of various implementation against our dataset of 130 experiments. We found that the EKF and UKF perform significantly better than the original system. By increasing model complexity, the elevation error is reduced while computational time is increased. We also found that both non-additive and additive noise models perform similarly well; however, the non-additive is much slower. The best performance method overall was shown to be the UKF15, which can reduce elevation error by as much as 63% compared to the original system. Finally, we demonstrated an example of introducing artificial constraints, such as flat-floor assumption, to the system and the resulting reduction in elevation error.

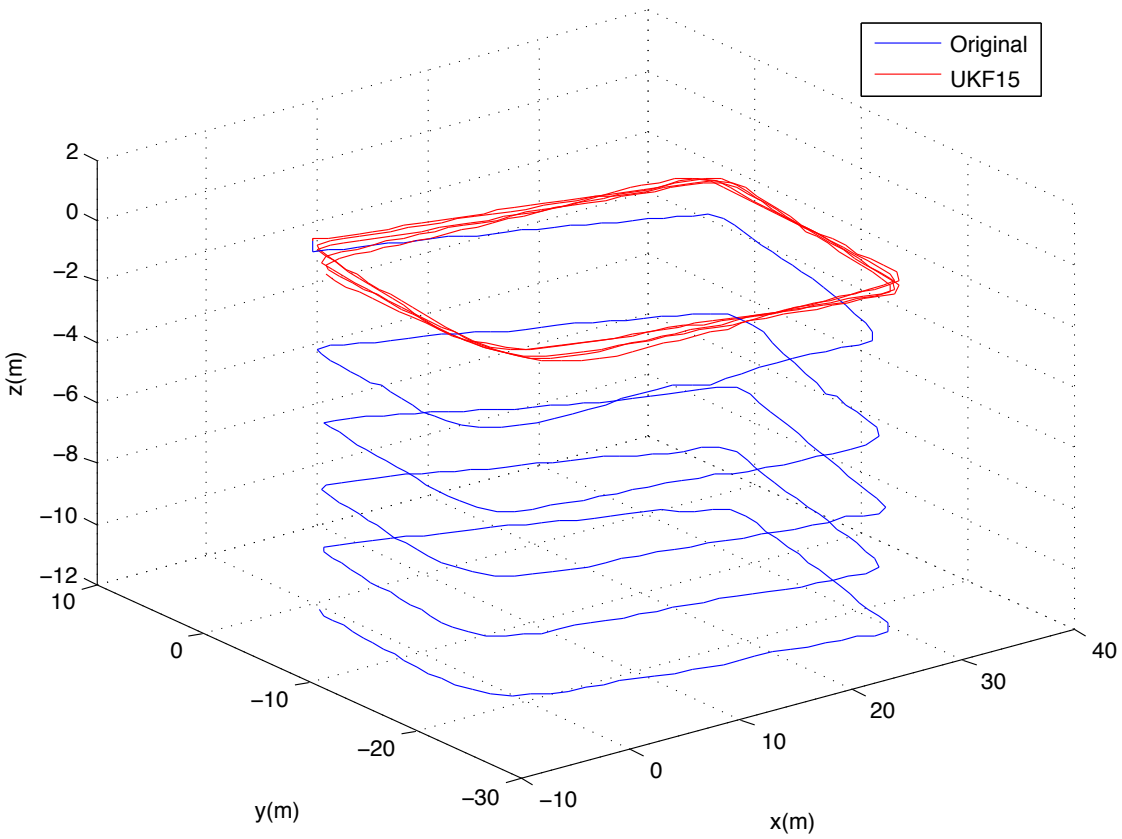

Figure 3.12: An example of the position output of original system and UKF15 of a subject walking in a rectagular path. We can see that the elevation error of UKF15 (red) is significantly less than the original system (blue).

CHAPTER IV

Magnetometer-Enhanced Personal Locator for Tunnels and GPS-Denied Outdoor Environments

4.1 Introduction

This chapter describes a system for estimating the position of walking persons in outdoor or underground environments in which a Global Positioning System (GPS) is not available and magnetic disturbances are few. The system is comprised of our earlier-developed Pedestrian Dead-Reckoning (PDR) system and a magnetometer-based software module which is fully discussed in *Ojeda and Borenstein* (2007a,b); *Borenstein et al.* (2009). For completeness, we provide a brief description of this module in Section 2.1.

The PDR system uses a foot-mounted Inertial Measurement Unit (IMU), which includes a three-axis gyroscope, a three-axis accelerometer, and a three-axis magnetometer. In our earlier work, we used only gyroscopes and accelerometers for position estimates and developed a method for correcting accelerometer drift that exploited the fact that an instrumented human foot briefly experiences zero velocity during each footfall. In addition, we developed a heuristic method for correcting heading errors caused by drift and other physical phenomena in the gyroscopes (*Borenstein et al.*, 2009). This heuristic method exploits the rectilinear features in man-made

structures. However, this method is useful only inside buildings that have rectilinear features.

4.2 Prior work

For most outdoor environments, GPS is the most accurate and cheapest solution (*Hofmann-Wellenhof et al.*, 1994) for all but the shortest of walks. For many military applications, however, it is desirable not to rely on GPS. In other applications, too, GPS may be unreliable due to occlusion (e.g., under dense tree canopies or in canyons). Moreover, in tunnels and caves, GPS is altogether unavailable. Local GPS-like beacons (also called pseudolites) can provide centimeter-accuracy over areas of several square miles, but the pseudolites have to be preinstalled before a mission and even they do not penetrate well into structures (*Cobb*, 1997).

Unlike gyros, which estimate relative changes in heading, the magnetometer is a sensor modality with the ability to estimate absolute heading. The earth's magnetic properties have been known and used in compasses for navigation for hundreds of years. Modern electronic magnetometers are used extensively in aviation and watercraft for measuring heading with respect to Earth's magnetic north. However, magnetometers can be affected by magnetic disturbances (as will be discussed in more detail), and are generally not reliable inside modern structures. Nonetheless, for natural outdoor environments, magnetometers are suitable for bounding the otherwise unlimited growth of heading errors derived from gyros. Much research on the use of magnetometers for personal positioning applications has been conducted in recent years. Some approaches use magnetometers exclusively for heading estimation (*Cho et al.*, 2003), while others integrate it tightly with an IMU (*Aparicio et al.*, 2004; *Yun and Bachmann*, 2006).

4.3 Magnetometer and Pedestrian Dead-Reckoning system

Since the Heuristic Drift Elimination (HDE) method introduced in *Borenstein et al.* (2009) cannot be applied to outdoor walking, another sensor modality is needed to counteract the effects of gyro drift. The magnetometer is a good choice because it works very well in many outdoor environments.

For the purpose of this chapter, we consider the PDR system to be a black box. This black box outputs position and heading of the walker in real-time and at footfall intervals. Also available at every instance of footfall are the ZUPT-corrected accelerometer readings. Since the instrumented foot is stationary during footfalls, we can exploit the fact that the only acceleration affecting the accelerometers is that of gravity. Consequently, we can determine roll and pitch, collectively called tilt of the IMU (which also houses the magnetometer) with considerable accuracy at the moment of footfall. Using tilt estimates based on accelerometers allows us to estimate and correct tilt errors.

The remaining problem is estimating heading errors. In order to simplify the discussion of this topic we introduce the term "virtual gyro". The output of the virtual z-axis gyro is the rate of turn around the z-axis. If we were able to estimate the virtual drift of the virtual z-axis gyro (which is, of course, the result of real drift in the three real gyros), then we could correct all real gyro drift errors, since we already have reasonably good estimates for the drift of x and y-axis gyros from the accelerometers. As such, for the remainder of this chapter we will limit our discussion to the problem of estimating virtual z-axis gyro drift.

One advantage of treating the PDR system as a black box, instead of trying to integrate the magnetometer with the full set of attitude/position equations, is that the proposed magnetometer functionality is modular and easy to port to other applications. Within the PDR system, the modular approach makes it easy to switch the magnetometer module on or off—a function that is useful, for example, when

switching to the HDE module when the user enters a building.

The main contributions of this chapter are:

- A practical calibration method for incorporating magnetometers to the PDR system that does not require the user to perform any explicit calibration procedure
- A robust real-time algorithm for sensor fusion and magnetic disturbance detection by comparing with IMU and the earth magnetic field model
- An evaluation using our real-world experiment dataset that demonstrates the performance of the algorithm, including results from a test conducted by firefighters

4.3.0.1 Earth magnetic field and the magnetometers

A three-axis magnetometer, such as the one built into the Memsense nIMU used in our PDR system, can determine the 3-dimensional direction of the magnetic field around the sensor. Absolute magnetic heading Ψ can be measured by decomposing the 3-dimensional magnetic field vector into three components, H_x, H_y, H_z, which are aligned with the navigation frame of the IMU.

$$\Psi = -\text{atan2}(\text{H}_\text{y}, \text{H}_\text{x}) \tag{4.1}$$

where H is the magnetic field in the IMU's navigation frame (the X-axis coincides with the magnetic north and the Y axis points east). The angle between magnetic north and geographic north is called magnetic declination angle which varies from location to location as shown in Figure 4.2.

The angle between the magnetic field H and the horizon is called magnetic inclination or magnetic dip which also varies with location on earth as shown in Figure 4.3.

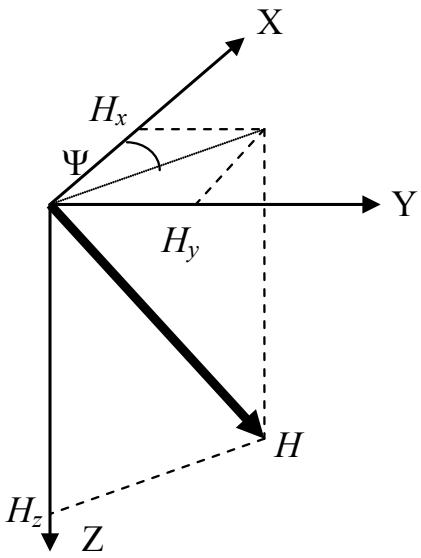

Figure 4.1: The graphical visualization of the Earth's magnetic field measurement H from magnetometers. Ψ is the heading with respect to the magnetic north.

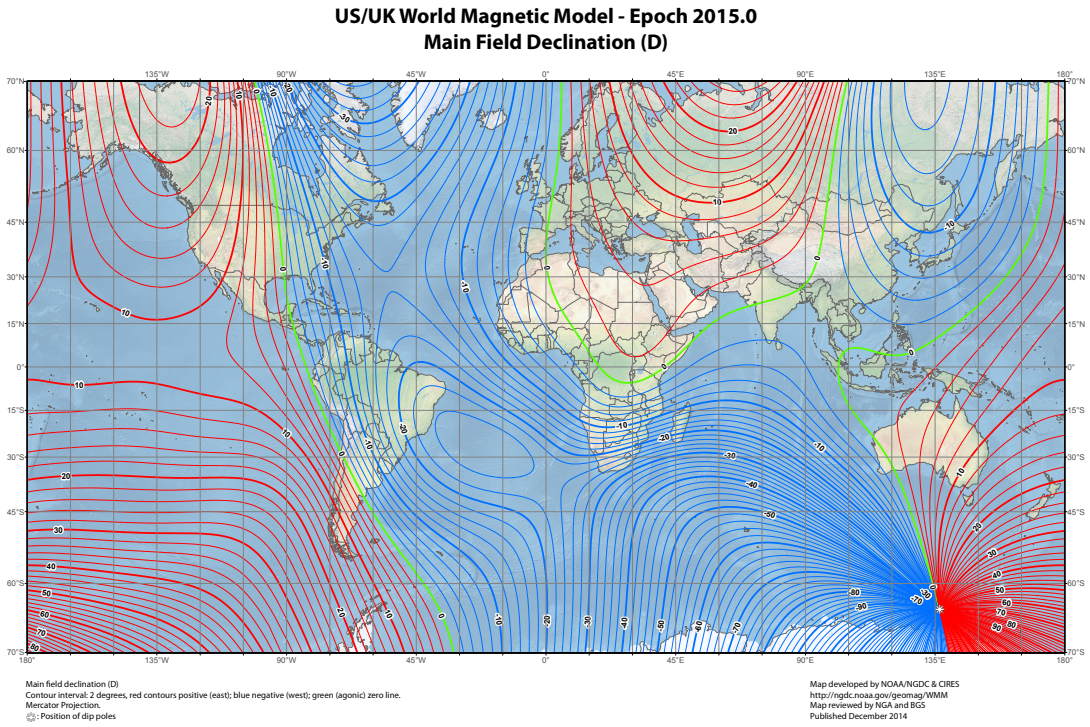

Figure 4.2: Magnetic declination angle map (*Maus et al.*, 2010)

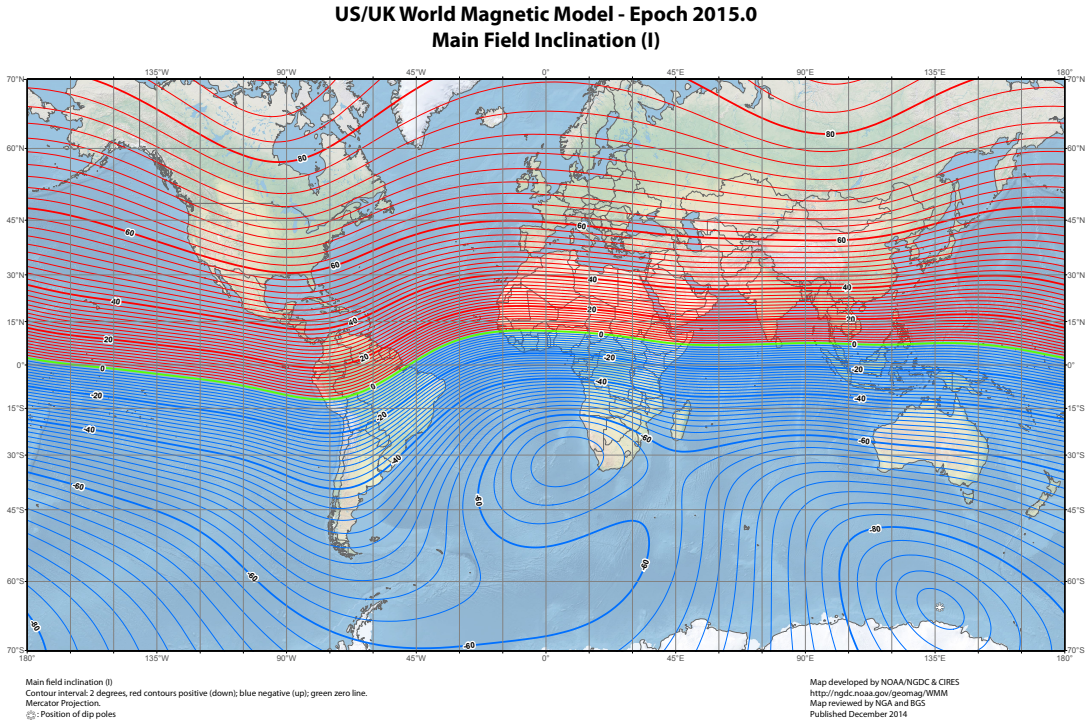

Figure 4.3: Magnetic inclination angle map (*Maus et al.*, 2010)

Since the measurement is taken in the IMU's body frame (where the X-axis points forward and the Y-axis points to the right of the sensor) the IMU's pose must be known or estimated so that the sensor reading in the body frame can be transformed into the navigation frame. Thus, any misalignment between the magnetometer component and the other sensors (notably the accelerometers, which estimate tilt at footfalls) will contribute to the heading calculation error.

Additional errors result from local magnetic disturbances, such as from electromagnetic fields around power lines, electronics, and steel structures. These errors can be divided into hard and soft iron errors. We only list these error sources below; a more exhaustive discussion is provided in (*Gebre-Egziabher et al.*, 2001). For all magnetometer applications, accuracy can be improved significantly by performing a calibration procedure prior to each trip as will also be discussed, below.

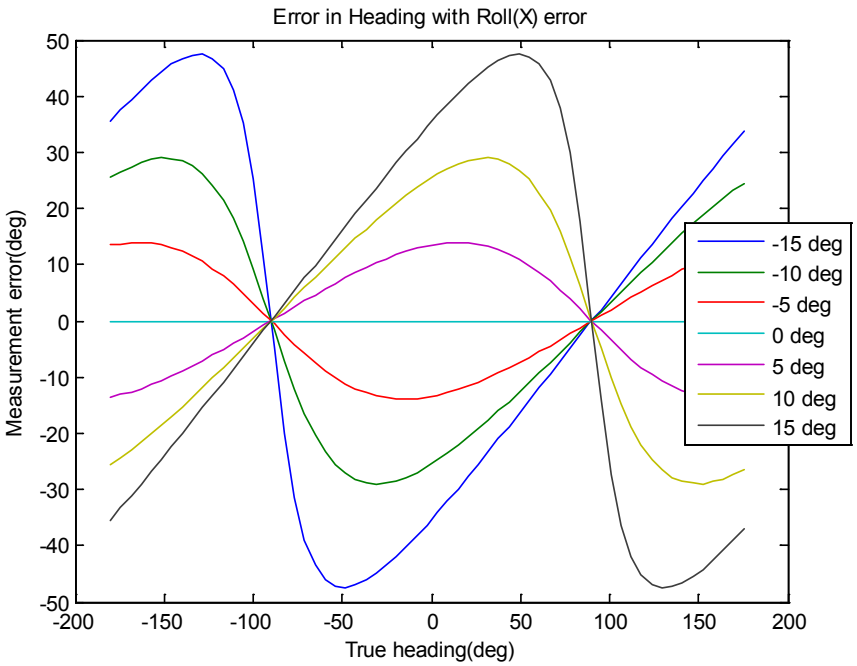

Figure 4.4: Heading measurement error due to misalignment in various roll angles

4.3.0.2 Typical errors with magnetometers

1. Hard Iron effect

   The hard iron effect is caused by nearby ferromagnetic materials that produce a constant additive magnetic field. This effect results in an offset in the sensor measurement independent of heading.

2. Soft Iron effect

   The soft iron effect is caused by materials that change the magnetic field depending on the orientation of the sensor. This error distorts measurements from the magnetometer as a function of sensor heading.

3. Misalignment

   To some degree, the different sensor modalities inside the IMU are misaligned relative to each other. This will result in crossover measurements in all sensor axes.

4.4 Calibration Procedure

The calibration procedure for the magnetometer is arranged into two steps. The first step is to eliminate sensor misalignment, while the second step aims at correcting hard and soft iron errors of the sensor.

4.4.1 Step 1 – Misalignment correction

When rotating the magnetometer horizontally, misalignment causes the magnetometer's Z-axis to be different from the world Z-axis. This causes the measured field vectors to be located on an inclined plane that corresponds to the misalignment angle. To remove this misalignment, we fit a plane to all measurements using a linear least square approach. Then, we rotate the plane so that the normal of the plane is parallel to the world Z-axis.

Equation of a plane:

$$aH_x + bH_y + cH_z + d = 0 \tag{4.2}$$

The normal to this plane is defined as:

$$n = \left[\frac{a}{\sqrt{a^2+b^2+c^2}}, \frac{b}{\sqrt{a^2+b^2+c^2}}, \frac{c}{\sqrt{a^2+b^2+c^2}}\right]^T \tag{4.3}$$

In order to rotate the normal vector to make it vertical, a rotation axis and rotation angle can be found by computing:

$$r = n \times [0,0,1]^T = [n_y, -n_x, 0]^T \tag{4.4}$$

$$\theta = \arccos\left(\frac{\mathrm{n}}{\|\mathrm{n}\|}\right) \tag{4.5}$$

By rotating all subsequence measurements through this angle along the rotation

axis, we can correct for the misalignment of the magnetometer.

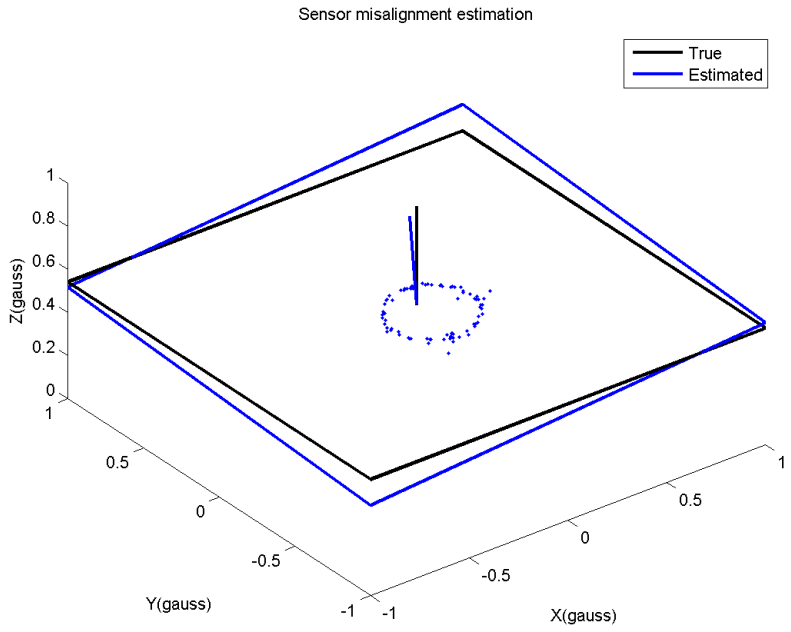

Figure 4.5: Misalignment estimation between magnetometer and accelerometer. The difference between the blue plane and the black horizontal plane is the misalignment between the two sensors

4.4.2 Step 2 - Soft iron and hard iron calibration

The second step of the calibration process exploits the fact that in a disturbance-free environment and after performing Step 1, a full rotation around the Z-axis should yield magnetic field vectors of identical magnitude. One can picture this condition as the locus of the tip of all measured field vectors. Since the magnitude of the horizontal magnetic field, H_{xy}, is constant, this locus should ideally be a circle.

$$H_x^2 + H_y^2 = (H_{xy}cos(\Psi))^2 + (H_{xy}sin(\Psi))^2 \tag{4.6}$$

$$= \|H_{xy}\|^2 \tag{4.7}$$

However, in practice, the hard and soft iron errors will distort that circle. Specifi-

cally, the hard iron disturbances move the center of the circle, and the soft iron effects distort the shape of the circle to that of an ellipse.

An overly optimistic assumption is that there is no crossover effect between the axes of the magnetometer (*Gebre-Egziabher et al.*, 2001). However, in practice, there is crossover interaction, and as a result the ellipse becomes tilted.

We can find the center of the tilted ellipse that has been displaced by the hard iron effects, by using a linear least square approach. The general equation for tilted ellipse with center at $[x_c, y_c]$ is:

$$a\left(x - x_c\right)^2 + b\left(y - y_c\right)^2 + c\left(x - x_c\right)\left(y - y_c\right) = 1 \tag{4.8}$$

By assuming $[x_c, y_c]$ are small, we can remove quadratic terms:

$$ax^2 + by^2 + cxy - (2a + c)x_c x - (2b + c)y_c y = 1 \tag{4.9}$$

Defining

$$d = -(2a + c) \qquad\qquad e = -(2b + c)$$

we can now rewrite the linear equation of the ellipse:

$$ax^2 + by^2 + cxy + dx + ey = 1 \tag{4.10}$$

By solving Equation (4.10) using linear least squares regression, the displacement of the center of the ellipse can be found:

$$x_c = \frac{d}{-(2a + c)} \qquad\qquad y_c = \frac{e}{-(2b + c)} \tag{4.11}$$

The equation for the same ellipse but with its center at the origin is:

$$ax^2 + by^2 + cxy = 1 \tag{4.12}$$

Equation (4.12) can be written in matrix form:

$$\begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} a & c/2 \\ c/2 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}^T M \begin{bmatrix} x \\ y \end{bmatrix} = 1 \tag{4.13}$$

In order to estimate the soft iron effect, we can decompose the matrix M using Singular Value Decomposition $M = U\Sigma V^*$. Then, the semi-major and semi-minor axes are the columns of the nearest orthonormal matrix of M, which is $A = UV^*$. The length of the ellipse is given by the diagonal components of Σ.

By combining the above estimations, we can correct subsequent measurements:

$$\begin{bmatrix} H_x^{corrected} \\ H_y^{corrected} \end{bmatrix} = A \begin{bmatrix} \frac{1}{\Sigma_{11}} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} A^T \begin{bmatrix} H_x - x_c \\ H_y - y_c \end{bmatrix} \tag{4.14}$$

After performing these corrections, one can estimate heading from Equation (4.1).

4.4.3 Robust estimation

In practice, apart from the local magnetic field, there are magnetic disturbances and noise that are also collected during calibration. Since the least square estimation of Eqs. (4.2) and (4.10) treats all samples equally, noise will add bias to the estimation. To cope with this problem, the maximum likelihood estimation can be used instead. In our implementation, we use the Huber estimation (*Huber*, 2011) with good success.

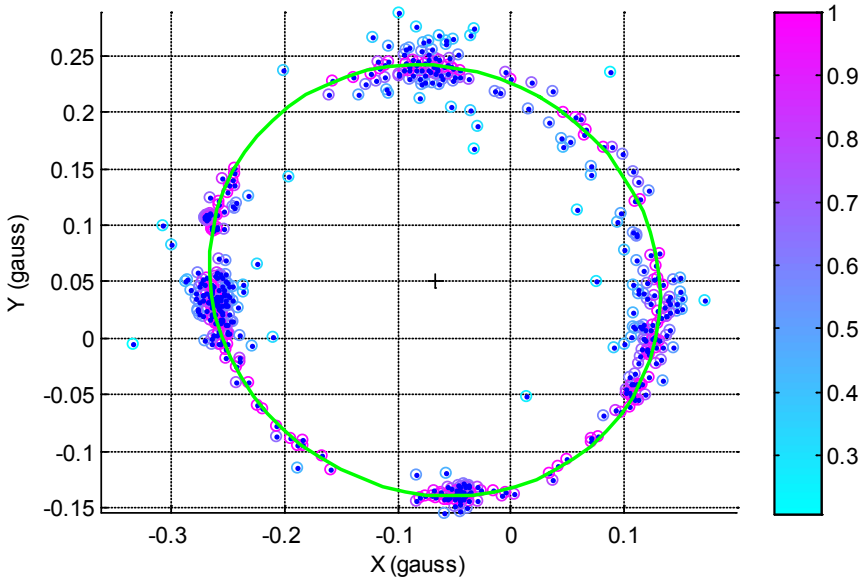

Figure 4.6: Result of soft/hard iron calibration with robust estimation. The blue dots are magnetic field measurements H_x, H_y from the magnetometers. The color of the small circles around each measurement represent the weight of each point from the Huber estimation. The black crosshair near the center shows the estimated displacement of the center due to the hard iron effect and the green ellipse shows the distortion of the ideal circle due to the soft iron effect.

4.4.4 Calibration procedure

For best results, a calibration of the magnetometer has to be performed once at the beginning of every walk. The calibration procedure requires the user to walk around in a circle for at least one full circle. The walk does not have to follow a circular contour with any accuracy; the only requirement is that the magnetometer experiences a full 360-degree rotation around the vertical axis. As a concession to practicality, we implemented the calibration process so that the system performs it as a background task. In practice, the user can start a walk anywhere, without performing the calibration circle immediately at the beginning of the walk. The PDR system will track the user immediately, albeit with the magnetometer calibrated with the calibration data from the previous walk. Then, when the user has an opportunity and a suitable magnetically undisturbed location, the user can perform the calibra-

tion circle and continue with the mission. The system automatically notices when a full circle has been completed, performs the mathematical operations needed for the calibration, and then uses the fresh calibration data from then on.

4.5 Real-time algorithm

4.5.1 Magnetic disturbance detection

During operation, especially inside modern buildings and in urban environments, a magnetometer is subject to many external disturbances. For example, since our magnetometer is mounted in the heel of the user's footwear, large metal objects such as manhole covers or underground pipes can create large disturbances. To minimize the effect of external magnetic disturbances, our system attempts to detect the presence of these temporary disturbances. Once detected, the affected magnetometer readings can be discarded and replaced by IMU-derived heading estimates.

To detect magnetic disturbances, we monitor two criteria:

1. Magnetic field strength

   If the measured field strength deviates by more than a predefined difference from the theoretical field strength based on the International Geomagnetic Reference Field (IGRF11) model (*Finlay et al.*, 2010) and World Magnetic Model (WMM2015) (*Maus et al.*, 2010), the measurement is flagged as a magnetic disturbance. However, in order to use the IGRF11/WMM2015 model, the local latitude and longitude have to be known as in Figure 4.7. In practice, we substitute for the theoretical IGRF11/WMM2015 model the average of measured field strengths during start up. This is reasonable since we work under the assumption that the system is initialized in an area that has only small magnetic disturbances. Moreover, the operating area of the the system is limited by human walking distance; thus, the local magnetic field strength can be assumed to

be constant. We found empirically that this average provides sufficient accuracy and it also eliminates the need for users to enter the local coordinates into the system.

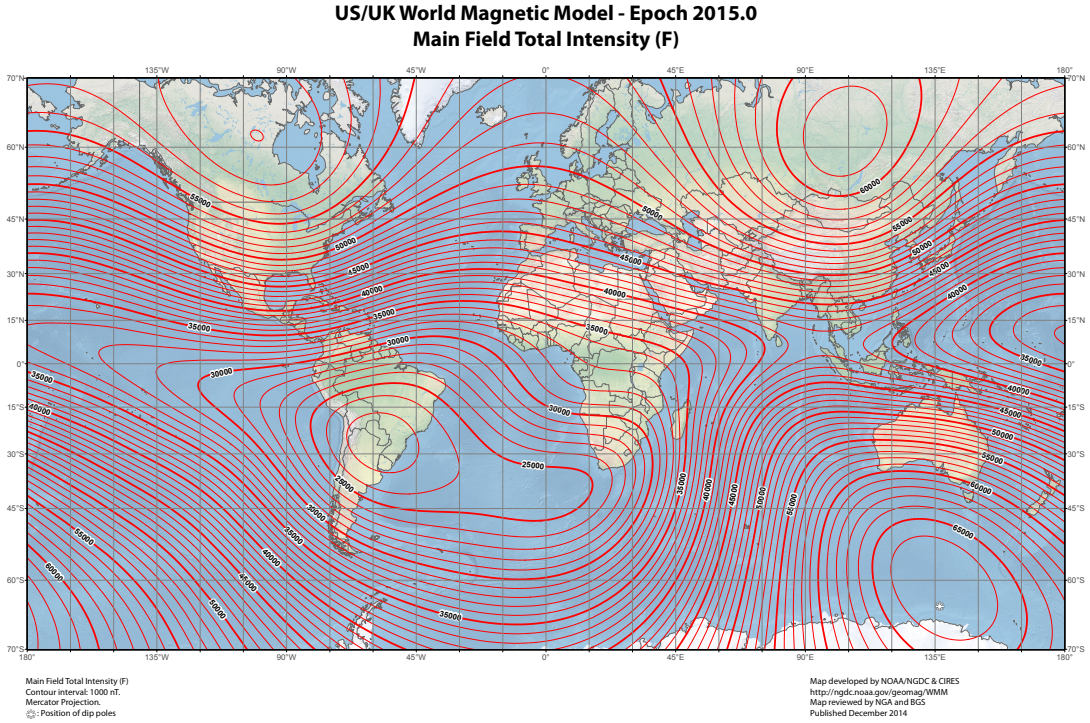

Figure 4.7: Magnetic field strength map (*Maus et al.*, 2010)

2. Rate of turn

   Since the rate of turn from gyroscopes has no effect due to magnetic disturbances, we compare the rate of turn derived from changes in the magnetometer heading to the rate of turn computed from the gyros in the PDR system. If they disagree, the magnetometer measurement is flagged as a disturbance.

For both criteria, once a disturbance is flagged the flag stays in effect for several subsequent magnetometer measurements. We chose this conservative approach because it is preferable to lose a few measurements rather than introduce erroneous magnetometer data into the system.

4.5.2 Sensor fusion

The position output from the IMU of the PDR system is not very noisy. However, heading errors grow without bound because of drift in the gyros. In contrast, heading derived from the magnetometer has no drift but has substantial noise due to external disturbance. To improve PDR position estimation, both measurements can be combined using a Kalman filter. The magnetic heading measurement that is not flagged as a disturbance is fed to the Kalman filter as described in the previous chapter.

In the worse case when the magnetic disturbance is presence everywhere, all the magnetic measurements will be discarded and the position output from the system will be the position calculated from IMU only. Thus, the performance of the system is equal or greater than the original system with IMU only.

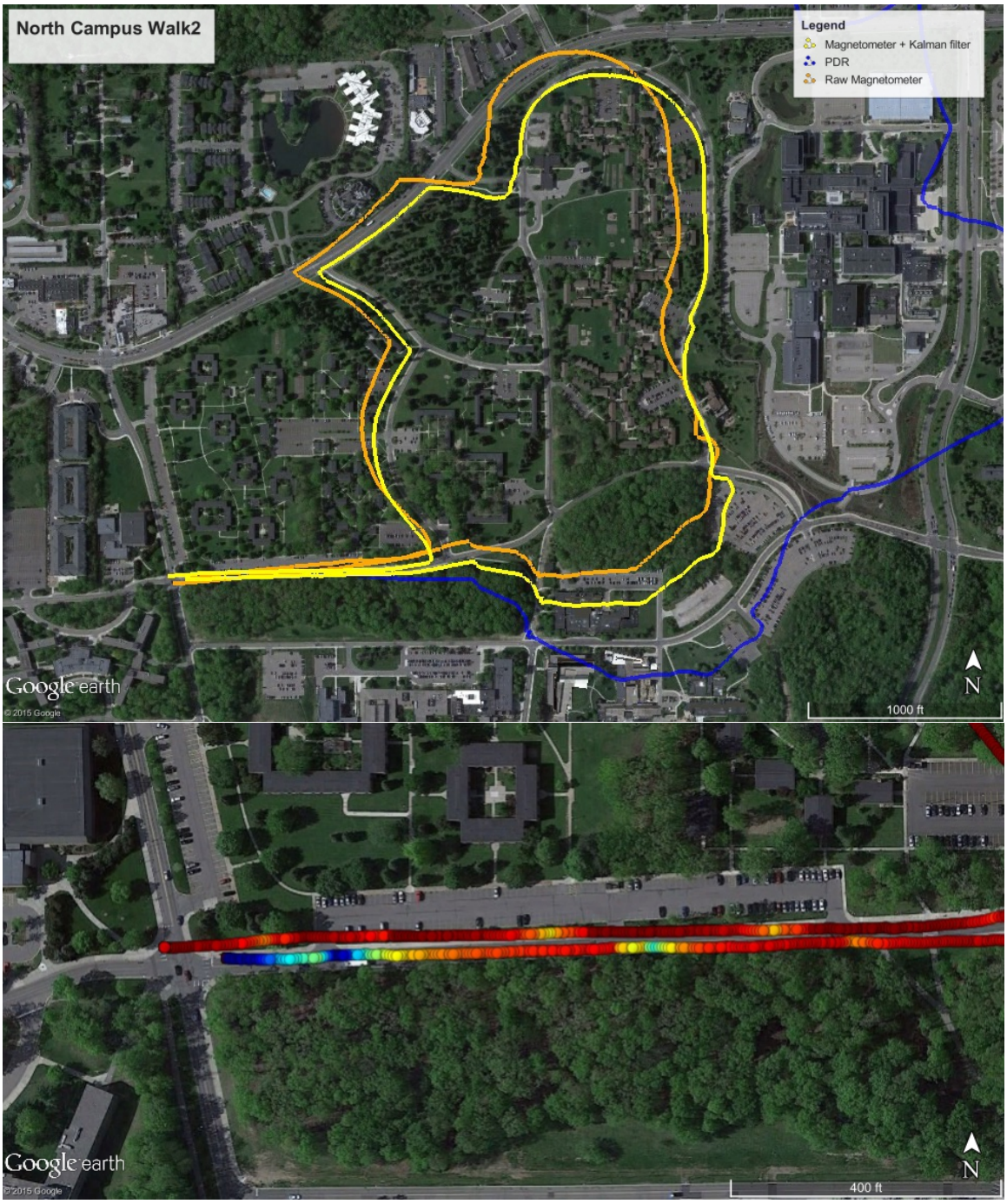

Figure 4.8: (Top) Trajectories output of a subject walked along sidewalks. The blue line is an errorneous PDR-only output using only an IMU. The orange line is an output using magnetic heading directly from magnetometer. The yellow line is an output of a Kalman filter combining the magnetometer with the PDR system. The bottom figure shows a section of the top figure with color varied by heading angle covariance. The red dots represent small covariance and blue for large covariance due to disturbances. The magnetic disturbances can be seen in several locations along the sidewalks.

4.6 Experimental results

4.6.1 North Campus walks

Several experiments were conducted around the University of Michigan's North Campus area by multiple subjects. The experiments started with a subject wearing a PDR system outside a building. Right after the recording began, the subject walked in circles for the magnetometer calibration procedure. After the system reported good calibration, the subject then walked around the campus and back to the starting position.

The total distance travelled for each experiment varied from 1 km to 5.4 km. All 33 experiments show that the position errors using the magnetometer are an average of 1.8% of the distance travelled; the PDR-only error can be much larger. There are a few experiments in which the PDR-only performs better than the magnetometer, but in these cases the position errors of both are comparably small. Figure 4.8 shows the result from one of the experiment in this dataset. The results for all North Campus experiments are shown in Table 4.1 and Figure 4.9.

Experiments	Total distance (m)	% position error per distance travelled	
		PDR Only	PDR+Magnetometer
nc_lawn_ccw	1379.31	7.62	1.89
nc_lawn_cw	1384.82	3.80	2.56
nw4_ccw	2611.88	2.87	1.31
nw4_cw	2623.79	37.34	2.83
nw5	1264.10	14.79	1.94
nw6	1300.75	4.88	1.36
nw7	1305.69	3.27	0.37
nw8	2785.73	7.26	0.81
hayward	1444.98	8.75	1.22
nc2	1851.63	76.30	2.06
nc3	2196.42	1.75	3.71
nc4	2514.31	2.82	3.18
nc_cmplx	3118.72	28.22	2.02
gps_walk3	1810.40	24.09	1.73
gps_walk4	3324.33	48.43	0.51
gps_walk5	1816.27	19.85	3.46
gps_walk6	1888.22	6.33	4.27
gps_walk7	1838.03	7.30	3.01
gps_walk8	4369.48	35.13	1.80
gps_walk9	5446.56	6.65	1.61
nc9_w1_walk1	3793.53	22.86	1.05
nc9_w2_walk1	3743.66	21.39	1.62
nc9_w1_walk2	960.99	6.69	1.61
nc9_w2_walk2	944.17	1.67	1.90
nc10_walk1	2244.01	21.21	1.88
st_walk2	2197.53	39.97	4.05
st_walk3	2277.42	12.65	2.63
st_walk4	2204.82	11.61	0.44
st_walk5	2480.00	5.87	0.67
st_walk6	1727.60	13.17	0.46
st_walk7	1703.16	1.97	0.94
st_walk10	1799.74	12.96	1.74
st_walk11	1577.03	0.22	1.64
	Mean	15.75	1.89
	95% CI	±5.64	±0.36
	Min	0.22	0.37
	Max	76.30	4.27

Table 4.1: Position error of North campus experiments. Without magnetometer, the PDR system has average position error as high as 15.8% while incorporating the magnetometer reduces average position error to 1.9%

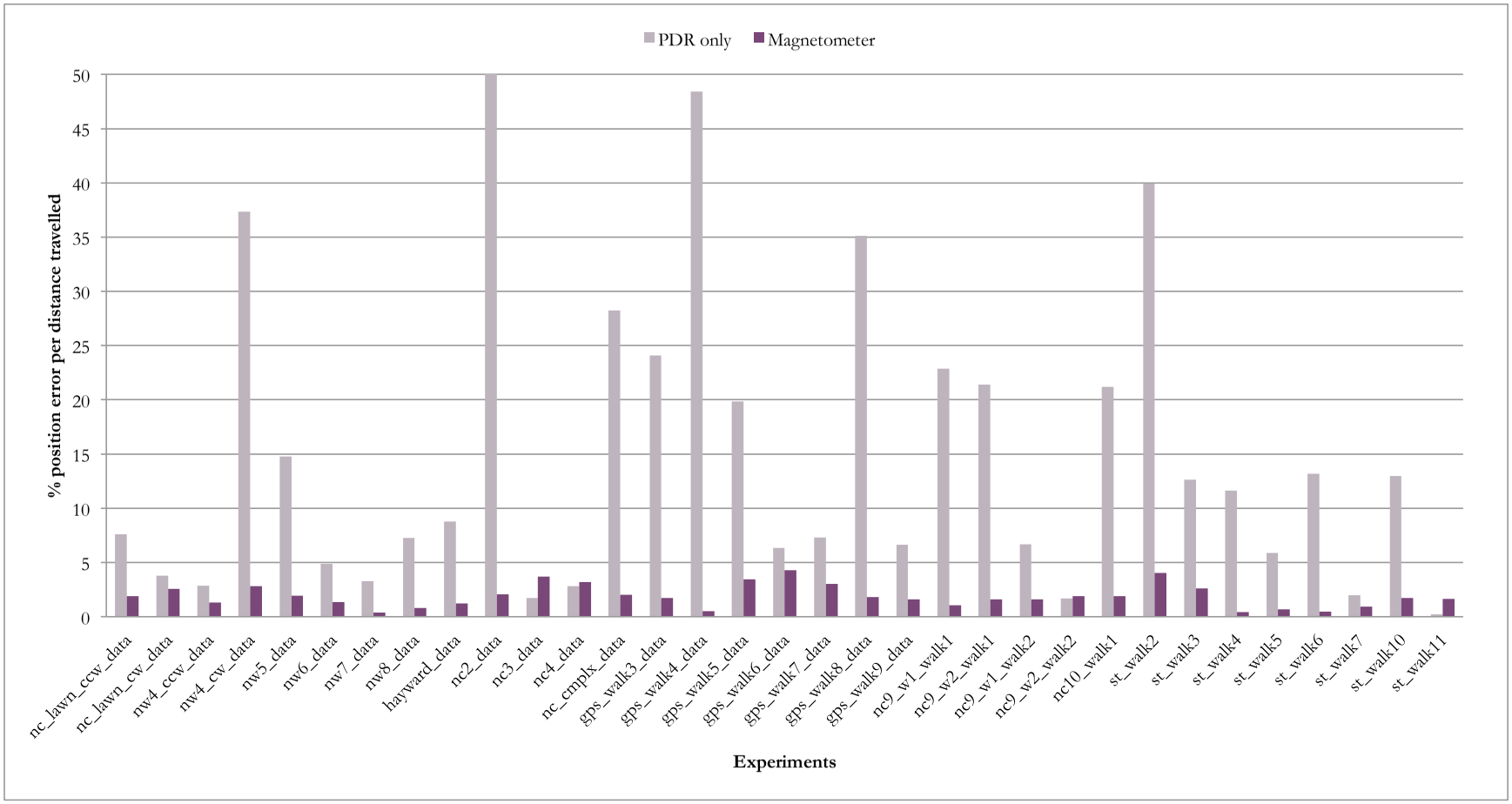

Figure 4.9: Performance comparision between PDR and magnetometer for all experiments.

4.6.2 Stone Wall Peak walks

The PDR system and its integrated magnetometer were tested by firefighters from the California Department of Forestry & Fire Protection (CALFIRE) under the patronage of the Center for Commercialization of Advanced Technology (CCAT) (*Kwanmuang et al.*, 2010, 2011). The test area was Stone Wall Peak, a mountainous hiking trail about 3.2 km (2.0 miles) long and located near San Diego, California, USA.

The nominal path, recorded by CALFIRE firefighters a few weeks ahead of the test (with a handheld GPS unit), is shown as the yellow line overlaid over the Google Earth satellite image of Figure 4.11. Two firefighters were wearing PDR systems with boots that were instrumented with IMUs. Their trajectories were displayed on an Operator Control Unit (OCU) in red and green colors, and, for simplicity, we refer to them as the red and green PDR systems or the red and green walkers, respectively. One additional experimenter walked the whole trail with a third PDR unit that used a strap-on IMU that produced blue trajectories.

The resulting trajectories, shown in Figure 4.12 and Table 4.2, suggest that none of the position errors of any one of the three tested PDR systems exceeded 45 meters ($\approx 1.4\%$ of distance traveled), and most of the time position errors were much smaller, less than 20 meters. These are rather remarkable results, given the length, ruggedness, and partially steep inclination of the terrain, as well as the duration of the hike of more than one hour in each direction. Indeed, for the most part, the performance of all three PDR units came close to that of consumer grade GPS units.

Figure 4.10: CALFIRE firefighters set up for a cover in place maneuver during the hike on Stone Wall Peak trail. The two firefighters with the red head gear wear our IMU-instrumented Altama boots. (*Kwanmuang et al.*, 2010)

Stone Wall Peak trail	
Total distance	≈3,200 m
PDR#1 (red)	
Position error	14.5 m (0.45%)
PDR#2 (green)	
Position error	45.79 m (1.43%)
PDR#3 (blue)	
Position error	47.56 m (1.48%)

Table 4.2: Position error recorded from three PDR units

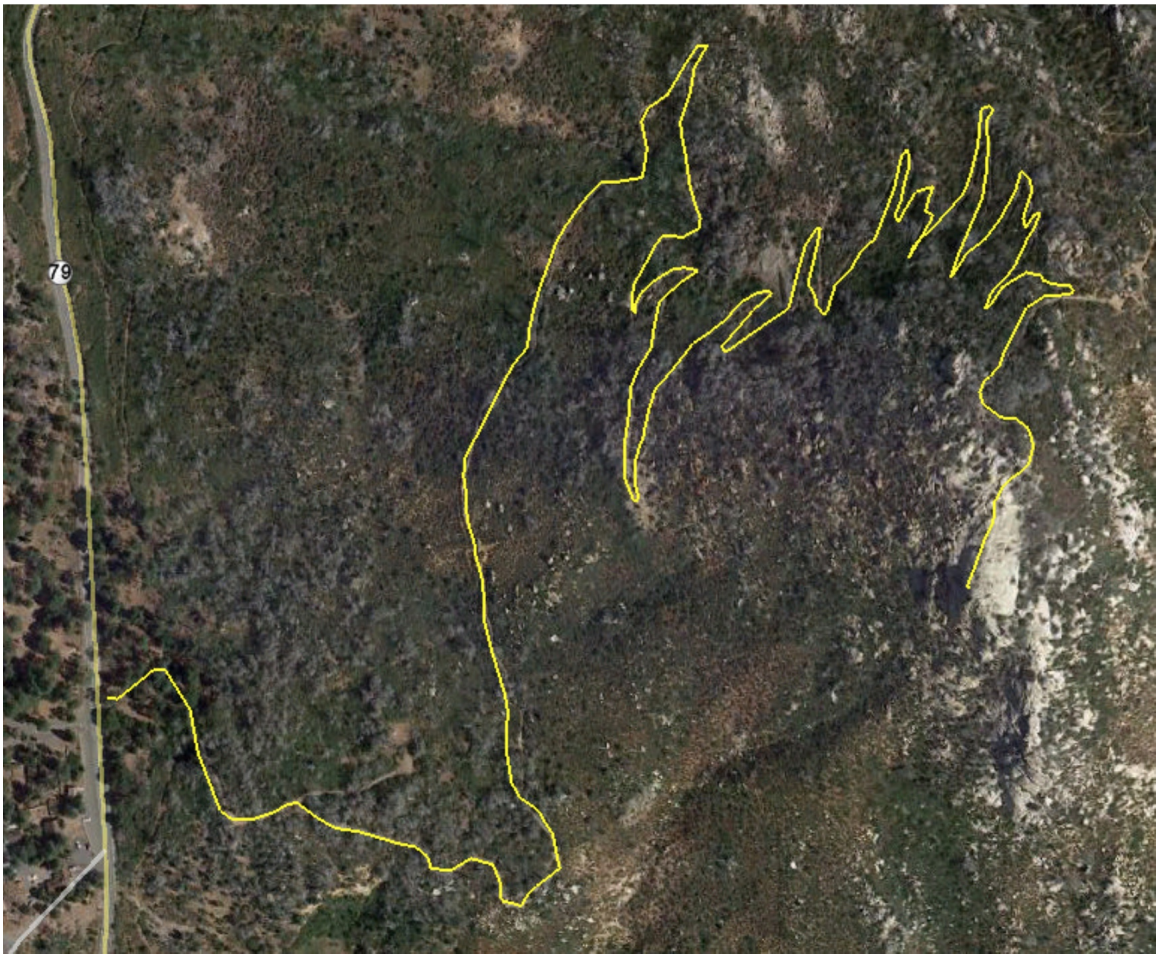

Figure 4.11: The Stone Wall Peak trail near Julian, California, was the test environment for the outdoor test on Day 3. The highlighted path was generated using a GPS device. (*Kwanmuang et al.*, 2010)

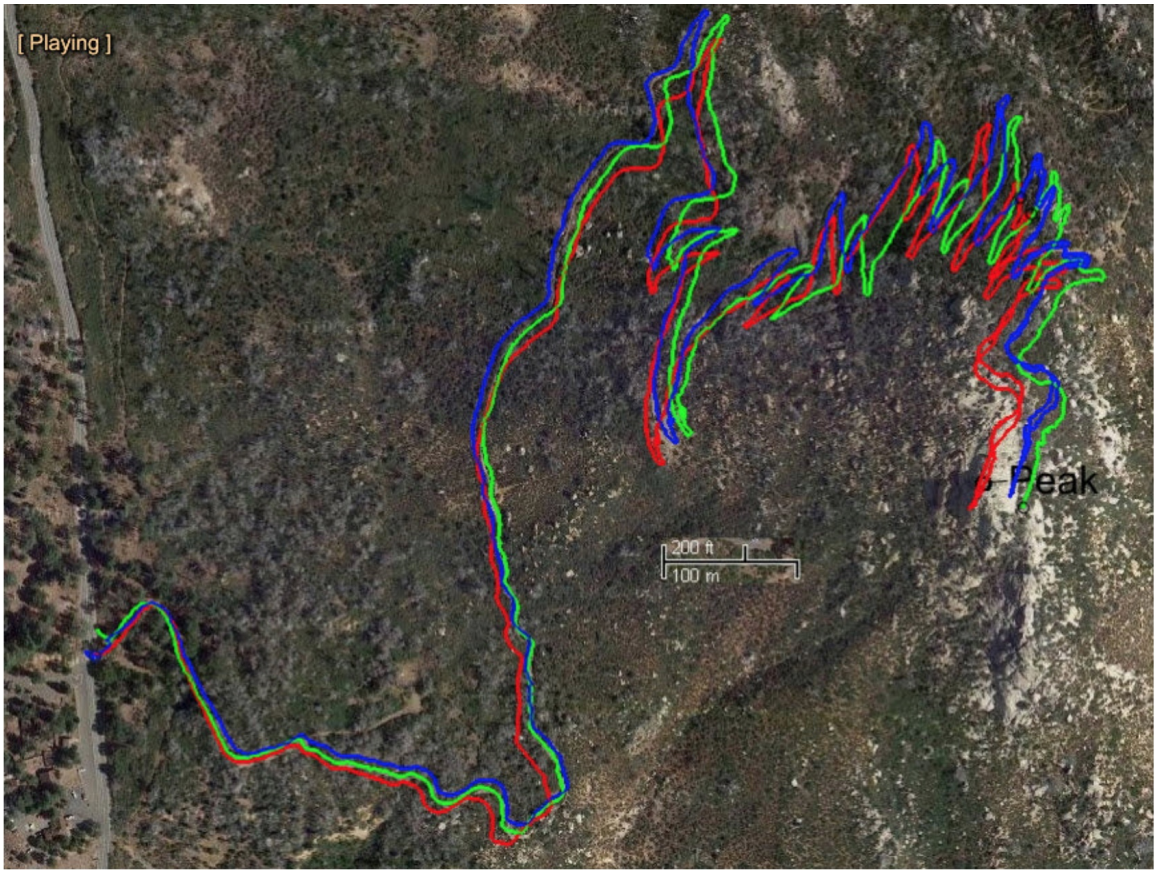

Figure 4.12: Trajectories as recorded by all three PDR units using IMU and magnetometer only, overlaid over a distortion-free 2-D satellite image from Google Maps. (*Kwanmuang et al.*, 2010)

4.7 Conclusion

This chapter introduced the magnetometer module of our personal locator system. We found that in a natural environment and after proper initial calibration, the magnetometer is capable of bounding heading errors from the IMU very effectively. In combination with our existing PDR system, which almost completely eliminates accelerometer drift by mounting the Memsense's nIMU in the heel of the user's boot, position errors are effectively limited to less than 1.9% of distance traveled. Trajectories recorded simultaneously by the three tested systems on three users show remarkable agreement with each other and demonstrate the robustness of the systems.

We conclude that with the help of the magnetometer module, our PDR system is capable of tracking the position of walkers on somewhat rugged terrain for periods of tens of minutes and more. To be clear, GPS is still the preferred sensor modality for outdoor hikes. However, in applications where GPS satellites may be temporarily occluded or the possibility exists for intentional GPS jamming, the magnetometer-enhanced PDR system can fill in with rather accurate position data during times of GPS outages.

CHAPTER V

Maximum-Likelihood Tracking for Pedestrian Dead-Reckoning System

5.1 Introduction

In this chapter, we propose a new method that utilizes a Pedestrian Dead-Reckoning (PDR) system (*Borenstein et al.*, 2009) on a human leader and a maximum-likelihood tracking system on a robot follower. This approach enables the system to work without a Global Positioning System (GPS) or any line-of-sight measurement. In cases where these measurements are available, however, they can be used to further improve the PDR system through intermittent recalibration.

The time between the moment the human leader starts walking and the moment the robot starts following can be anywhere from a few minutes to multiple hours. Once the leader starts walking, the PDR system onboard the leader transmits the leader's poses to the robot. The robot then stores these poses into its database to be used when it begins following. After the robot has started the mission, the odometry sensor and laser scanner onboard the robot are used to estimate the current robot pose and to map the obstacles around the robot using a Simultaneous Localization And Mapping (SLAM) algorithm (*Olson et al.*, 2006). The robot can then use this map to determine the trajectory that the human leader has taken.

One novelty of this method is that it is the first application we know of in which one agent completes the trajectory estimation of another agent while generating its own map of the environment.

In the following sections, we present a method of tracking another agent's position through the robot's map using Maximum-Likelihood (ML) estimation and Stochastic Gradient Descent (SGD)(*Robbins and Monro*, 1951). Moreover, we expand the method to be able to track multiple hypotheses in the case of ambiguity in surroundings.

The main contributions of this chapter are:

- Several methods for tracking: one based on a particle filter, the others based on maximum likelihood optimization using stochastic gradient descent
- An extension to the maximum-likelihood tracking for enabling the algorithm to track multiple hypotheses
- An evaluation that demonstrates the performance of the algorithms

5.2 Prior work

Current studies on leader-follower problems use GPS as the primary method for the robot to follow the human (*Teck Chew et al.*, 2004). By comparing the robot's current GPS coordinates with the human's coordinates, the robot can follow the leader without line-of-sight requirements. This approach, however, relies on the availability of a GPS signal that may become degraded due to obstructions.

Another prevalent method is to use stereovision on both the leader and the robot (*Naroditsky et al.*, 2009). In this scenario the leader sends image features from their camera to the robot to store in its database. The robot compares its current image features with the database to estimate the relative position of the human then tries to move to that position. This method can easily fail, however, if what was the scene

for the leader changes and looks different to the robot, or if the scene has too few distinct features to match. It is also bandwidth intensive.

The leader-follower problem can also be treated as a multiple robot localization problem (*Fox et al.*, 2000) in which both the leader and the follower use their sensors to estimate the location of the other. In this case, however, because the leader is a person rather than a robot, mounting sensors such as a laser scanner is not very practical.

Other researchers have used inertial measurement sensors on human leaders (*Woodman*, 2010; *Beauregard*, 2009) and particle filters exploiting a prior map. In this work, we eliminate the need for a prior map.

5.3 Background

5.3.1 Relative poses

The poses from the PDR system are formed by composing the changes measured by inertial measurement sensors. Thus, if we modify one pose, all following poses along the chain are projectively affected.

We can define state variables as relative poses y_i between human poses x^h_{i-1} and x^h_i. This variable is initialized to be a function of odometry input $y^0 = f(o)$:

$$y_i = [\tilde{x}_i, \tilde{y}_i, \tilde{\theta}_i,]^T \tag{5.1}$$

Therefore, a transformation $T(y_i)$ which transforms a pose x_{i-1}^h to x_i^h is given as:

$$x_i^h = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_i^h = T(y_i) \otimes x_{i-1}^h \tag{5.2}$$

$$x_i^h = \begin{bmatrix} \cos(\theta_{i-1})\tilde{x}_i - \sin(\theta_{i-1})\tilde{y}_i + x_{i-1} \\ \sin(\theta_{i-1})\tilde{x}_i + \cos(\theta_{i-1})\tilde{y}_i + y_{i-1} \\ \theta_{i-1} + \tilde{\theta}_i \end{bmatrix} \tag{5.3}$$

Thus, the pose at time t can be calculated from a function:

$$x_t^h = T(y_t) \otimes T(y_{t-1}) \otimes .... \otimes T(y_1) \otimes x_0^h \tag{5.4}$$

Given all relative poses $y_{0:n}$, the whole trajectory $x_{0:n}^h$ can also be obtained in the same way $x_{0:n}^h = \{x_0^h, x_1^h, ..., x_n^h\}$.

5.4 Method

We present methods for tracking a human leader in the following sections. First we discuss the human presence probability given obstacle detections. Second, a particle filter algorithm for tracking is given. Next, we present a Stochastic Gradient Descentfor Maximum Likelihood tracking (SGD_ML) and its variant (SGD_ML_Z). One limitation of SGD_ML and SGD_ML_Z is that they are only able to optimize poses to a single local minima and thus are unable to consider multiple hypotheses. In section 5.7, we propose a method that allows multiple trajectory hypotheses.

5.4.1 Human presence probability

Unlike other localization methods that utilize sensors to detect surroundings, obstacles, or structure features around the system to localize, the PDR system only estimates position and orientation of the human subject without knowing about the surrounding area. Moreover, the output from the PDR is corrupted by noise in the inertial measurement sensors.

The follower-robot can utilize its sensors to generate a map around itself. We assume that the environment did not change between when the human was there in the past and when the robot is following. The robot applies its knowledge of the environment to the leader's trajectory and estimates the probable path that the human took.

Using the map from the robot, we can estimate the likelihood that a human passed through that position using a "human presence probability" map. First, we assume that the human will not be present at the same place as obstacles; for example, a human will not walk through walls. Second, the further away from an obstacle a point is, it is increasingly likely that a human will be present in that location. Far away from obstacles, the human is equally probable to be present anywhere. We can think of this problem from another perspective. We can conceive of the human leader as a short-range obstacle detector that is always in free space. The closer this detector is to any obstacles, the more unlikely this detector would be there. In other domains, a different human presence probability map may be more appropriate. Our method can work with any given map.

From these assumptions, we can construct a human presence probability based on obstacle detection, as in Figure 5.1, where the shaded area on the right is the closest obstacle. For the area closest to the obstacle (area 1, normally within half a shoulder's breadth), human presence is highly improbable. Further away, as in area 2, we use a normalized Gaussian distribution $N(l, \sigma)$.

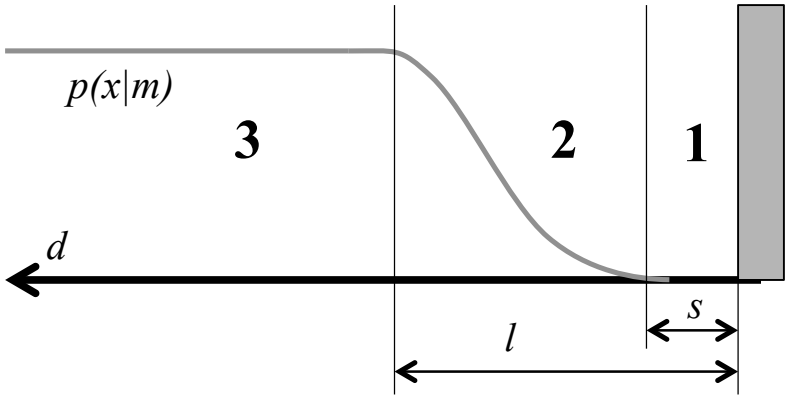

Figure 5.1: Human presence probability, $p(x|m)$, as a function of distance to the closest obstacle, d. The shaded area to the right is the obstacle. The curve shows the probability of human presence at each distance from the closest obstacles.

This distribution reflects the behavior that a human leader is unlikely to stay close to an obstacle and is increasingly likely to be further away. By setting $\sigma \ll l$, the continuity at $d = s$ is maintained. Area 3 is where the obstacle is far away and human presence is most plausible.

Others have observed distributions like this from empirical data (*Fajen and Warren*, 2003). Specifically, our model is:

$$p(x|m_{ML}) = \begin{cases} 0 & d < s \\ \exp(-(d-l)^2/2\sigma^2) & s < d < l \\ 1 & d > l \end{cases} \tag{5.5}$$

where d is the distance to closest obstacle. We can precompute this probability for each pose inside the map for a given occupancy map using a distance transform and equation (5.5). An example of an input occupancy map and resulting human presence probability map is in Figure 5.2.

Figure 5.2: Example of human presence probability map. (Left) Occupancy map from a simulated corridor with a divider in the middle (white – free space, black – obstacles). (Right) human presence probability map (white – high probability, black – low probability)

5.4.2 Graphical model of maximum likelihood tracking

Our goal is to estimate the trajectory of the human given both odometry data from the human and measurement data acquired by the robot. That is, we want $p(x^h|u, z, o)$ where x^h is the human trajectory, u is the robot's odometry input, z is robot laser scanning measurement and o is PDR odometry input.

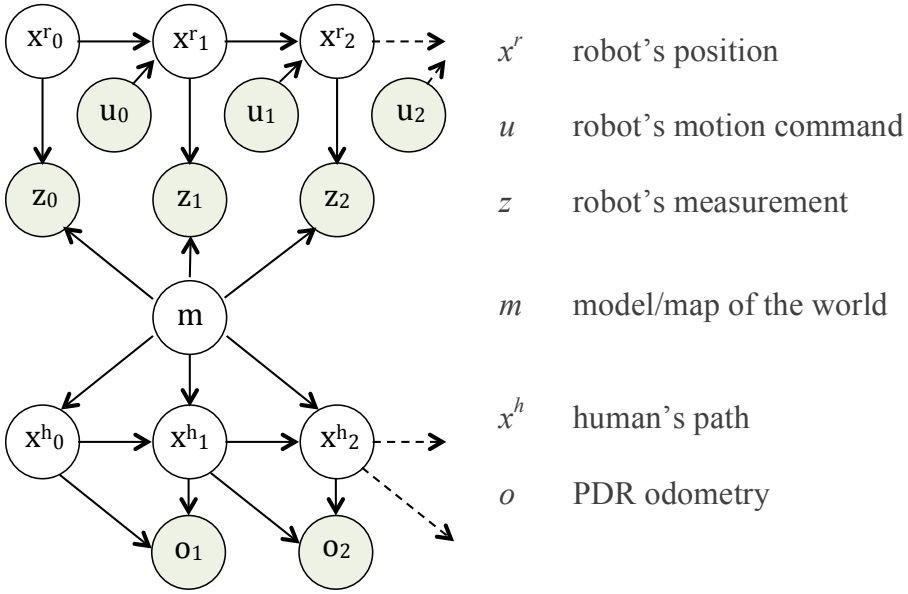

Figure 5.3: A probabilistic graphical model of human/robot trajectory. The top part of the graph shows the robot trajectory and its measurements of the map. The bottom part are a path of the human and the PDR odometry measurements of the path. The shaded variables are observed.

The posterior of the human trajectory $p(x^h|u,z,o)$ can be written as:

$$p(x^h|u,z,o) = \int p(x^h,m|u,z,o)dm \tag{5.6}$$

$$\text{(Bayes' rule)} \propto \int p(u,z,o|x^h,m)p(x^h,m)dm \tag{5.7}$$

$$\text{(Product rule)} = \int p(o|x^h,m,z,u)p(z,u|x^h,m)p(x^h,m)dm \tag{5.8}$$

The term $p(o|x^h,m,z,u)$ can be applied conditional independence and Bayes' rule on the $p(z,u|x^h,m)$ yields:

$$p(x^h|u,z,o) \propto \int p(o|x^h)p(x^h,m|z,u)dm \tag{5.9}$$

$$\text{(Product rule)} = \int p(o|x^h)p(x^h|m,z,u)p(m|z,u)dm \tag{5.10}$$

$$\text{(Cond ind.)} = \int p(o|x^h)p(x^h|m)p(m|z,u)dm \tag{5.11}$$

As described in section 5.4.1, the human presence probability map m_{ML} is obtained from the map posterior $p(m|z,u)$ generated from the robot.

$$p(x^h|u,z,o) \approx p(o|x^h)p(x^h|m_{ML}) \tag{5.12}$$

From the graphical model assumptions in Fig.5.3, the posterior $p(x^h|u,z,o)$ can be written as:

$$p(x^h|u,z,o) = \eta \prod p(o_i|x^h_{i-1},x^h_i) \prod p(x^h_i|m_{ML}) \tag{5.13}$$

5.5 Particle filter tracking

We first implemented particle filter tracking (Monte Carlo localization) (*Fox et al.*, 2001). MCL has been used widely in localization problems due to the effortlessness of implementation and it works well in many situations. MCL can also handle multi-

modal posteriors.

From Equation (5.13), using Bayes's rule:

$$p(x^h|u,z,o) \propto \prod p(x_i^h|x_{i-1}^h,o_i) \prod p(x_i^h|m_{ML}) \tag{5.14}$$

We can use the particle filter and sample from a distribution $p(x_i^h|x_{i-1}^h,o_i)$. The weight of each particle becomes:

$$w_i = p(x_i^h|m_{ML}) \cdot w_{i-1} \tag{5.15}$$

Although MCL works well in many scenarios, it can suffer from particle depletion, a situation in which there are not enough particles to accurately represent the distribution. This particle depletion problem is depicted in Figure 5.4. Increasing the number of particles can delay the onset of particle depletion, but with increasing computation cost.

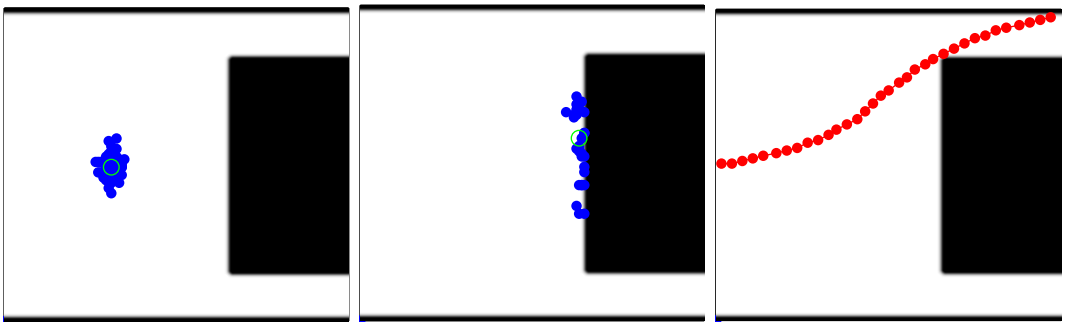

Figure 5.4: Particle depletion problem of monte-carlo localization. The human walked from left to right. Blue dots are particles representing human poses. (Right) Particles are all depleted while our maximum-likelihood algorithm (SGD_ML in red) is still able to track the human trajectory.

5.6 Maximum-Likelihood tracking

To avoid particle depletion in the MCL algorithm, we propose a new tracking algorithm using maximum-likelihood estimation. Since maximum-likelihood tracking

only keeps track of one trajectory, it is very memory efficient compared to a particle filter that has to keep track of all the particles.

The closed-form solution of the posterior through induction is given as:

$$p(x^h|u,z,o) = \eta p(x_0^h)p(x_0^h|m_{ML}) \prod_{i=1}^{n} [p(x_i^h|m_{ML})p(o_i|x_{i-1}^h, x_i^h)] \tag{5.16}$$

A cost function or a negative log likelihood of the posterior can be calculated by taking the logarithm of the equation (5.16):

$$\begin{aligned} \log(p(x^h|u,z,o)) = const + \log(p(x_0^h)) + \log(p(x_0^h|m_{ML})) + \\ \sum_{i=1}^{n} [\log(p(x_i^h|m_{ML})) + \log(p(o_i|x_i, x_i))] \end{aligned} \tag{5.17}$$

The human initial position x_0^h is assumed to be a Gaussian distribution $N(0, \Sigma_0)$. Similarly, the PDR odometry probability $p(o_i|x_{i-1}, x_i) = p(o_i|y_i)$ is assumed to be Gaussian $N(y_i^0, \Sigma_i)$ where $y^0 = f(o)$.

$$p(u_i|x_{i-1}, x_i) = \eta \exp\left(-\frac{1}{2}(y_i - y_i^0)^T \Sigma_i^{-1} (y_i - y_i^0)\right) \tag{5.18}$$

By inspection, the initial pose $p(x_0^h)$ is a special case of the above equation, where $y_0 = x_0^h$ and $y_0^0 = 0$:

$$p(x_0) = \eta \exp\left(-\frac{1}{2} x_0^T \Sigma_0^{-1} x_0\right) \tag{5.19}$$

$$p(x_0) = \eta \exp\left(-\frac{1}{2}(y_0 - 0)^T \Sigma_0^{-1} (y_0 - 0)\right) \tag{5.20}$$

We now define $d_i \equiv y_i - y_i^0$. The cost of the PDR odometry update becomes:

$$\sum_{i=0}^{n} \log(p(o_i|x_{i-1}^h, x_i^h)) = -\frac{1}{2}\sum_{i=0}^{n}(y_i - y_i^0)^T \Sigma_i^{-1}(y_i - y_i^0) \tag{5.21}$$

$$= -\frac{1}{2}\sum_{i=0}^{n}(d_i)^T \Sigma_i^{-1}(d_i) \tag{5.22}$$

$$= -\frac{1}{2} d^T \Sigma^{-1} d \tag{5.23}$$

where $d = [d_0, d_1, ..., d_n]$

As for $\log(p(x_i^h|m_{ML}))$, the probability of the poses given a map can be easily looked up in the pre-generated human presence probability map (map_{ML}), where x_i^h is inside that cell.

$$p(x_i^h|m_{ML}) \approx h(x_i^h) = map_{ML}(i,j) \quad x_i^h \in \text{cell}(i^{th}, j^{th}) \tag{5.24}$$

$$\log(p(x_i^h|m_{ML})) = \log(h(x_i^h)) \tag{5.25}$$

As a result, the negative log-likelihood of the posterior and the cost function becomes:

$$J = -\log(p(x^h|u, z, o)) \tag{5.26}$$

$$J = \sum_{i=0}^{n} \big(-\log(h(x_i^h))\big) + \frac{1}{2} d^T \Sigma^{-1} d + const \tag{5.27}$$

Intuitively, we can see that the cost function is a function of the deviation from the odometry $d^T \Sigma^{-1} d$ and negative cost of the probability of the human can be presence at a given pose $-\log(h(x_i^h))$. If the pose deviates too far from the odometry measurement, the cost will increase as a function of the information matrix Σ^{-1}. A pose that has high certainty (small Σ), will incur a larger cost than the uncertain one.

If the human pose x_i^h is close to obstacles and unlikely to be present there,

$p(x_i^h|m_{ML})$ approaches zero, and a substantial cost will be incurred compared to poses further away from obstacles.

5.6.1 Stochastic Gradient Descent (SGD)

To find the poses that have lowest cost function, Stochastic Gradient Descent (SGD) can be used as an optimization solver. SGD optimizes each pose one at a time to reach the cost minima. We use SGD due to its robustness to local minima.

From Equation (5.27), The cost function for a single pose is:

$$J_i = -\log(h(x_i^h)) + \frac{1}{2(n+1)} d^T \Sigma^{-1} d \tag{5.28}$$

Using the chain rule, the gradient of the cost function for a single pose is:

$$\nabla J_i = -\frac{1}{h(x_i^h)} \cdot \frac{\partial h(x_i^h)}{\partial x_i^h} \cdot \frac{\partial x_i^h}{\partial y} \cdot \frac{\partial y}{\partial d} + \frac{1}{n+1} d^T \Sigma^{-1} \tag{5.29}$$

The first term, $h(x_i^h)$ can be easily looked-up from the human presence probability map as in equation (5.24). Next, $\partial h(x_i^h)/\partial x_i^h$ is the gradient of the map around x_i^h. The term $\partial x_i^h/\partial y$ can be calculated from the Jacobian matrix.

$$\frac{\partial x_i^h}{\partial y} = \begin{bmatrix} \frac{\partial x_i^h}{\partial y_0} & \frac{\partial x_i^h}{\partial y_1} & \cdots & \frac{\partial x_i^h}{\partial y_n} \end{bmatrix}_{2 \times 3n} \tag{5.30}$$

where each term $\frac{\partial x_i^h}{\partial y_j}$ is in the following form, for $j = 0$:

$$\frac{\partial x_i^h}{\partial y_0} = \begin{bmatrix} 1 & 0 & \sum_{k=0}^{i-1} \left(-\sin(\theta_k)\tilde{x}_{k+1} - \cos(\theta_k)\tilde{y}_{k+1}\right) \\ 0 & 1 & \sum_{k=0}^{i-1} \left(\cos(\theta_k)\tilde{x}_{k+1} - \sin(\theta_k)\tilde{y}_{k+1}\right) \end{bmatrix} \tag{5.31}$$

for $j > 0$ and $j \leq i$:

$$\frac{\partial x_i^h}{\partial y_j} = \begin{bmatrix} \cos(\theta_{j-1}) & -\sin(\theta_{j-1}) & \sum_{k=j}^{i-1} \left(-\sin(\theta_k)\tilde{x}_{k+1} - \cos(\theta_k)\tilde{y}_{k+1}\right) \\ \sin(\theta_{j-1}) & \cos(\theta_{j-1}) & \sum_{k=j}^{i-1} \left(\cos(\theta_k)\tilde{x}_{k+1} - \sin(\theta_k)\tilde{y}_{k+1}\right) \end{bmatrix} \tag{5.32}$$

for $j > i$:

$$\frac{\partial x_i^h}{\partial y_j} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.33}$$

Lastly, since we define $d \equiv y - y^0$, the term $\partial y / \partial d = 1$.

We can define the learning rate λ which will dictate the step size of the state correction. In addition, the state estimate moves in the opposite direction of the gradient (5.29). As a result, the state correction becomes:

$$d = \lambda \left(\frac{1}{h(x_i^h)} \cdot \frac{\partial h(x_i^h)}{\partial x_i^h} \cdot \frac{\partial x_i^h}{\partial y} - \frac{1}{n+1} d^T \Sigma^{-1} \right) \tag{5.34}$$

We use a learning rate that is harmonically-decreasing $\lambda = 1/t$, as in (*Robbins and Monro*, 1951). We found that if the initial learning rate λ_0 is too large, the state will move in large steps and is likely to jump around the solution. The worst case scenario is that it may jump and be trapped outside the convex area of the cost function, and thus fail to converge. On the other hand, smaller λ_0 will slow the convergence down and the solution may not converge in a reasonable time.

In summary, the SGD_ML method is given in the Algorithm 5.1. A simulation result of the SGD_ML algorithm at different iterations is shown in Figure 5.5.

One interesting aspect of the SGD_ML method is that, without any information from the map, the optimization result will be identical to the odometry path. This is because we do not have any additional information other than the odometry.

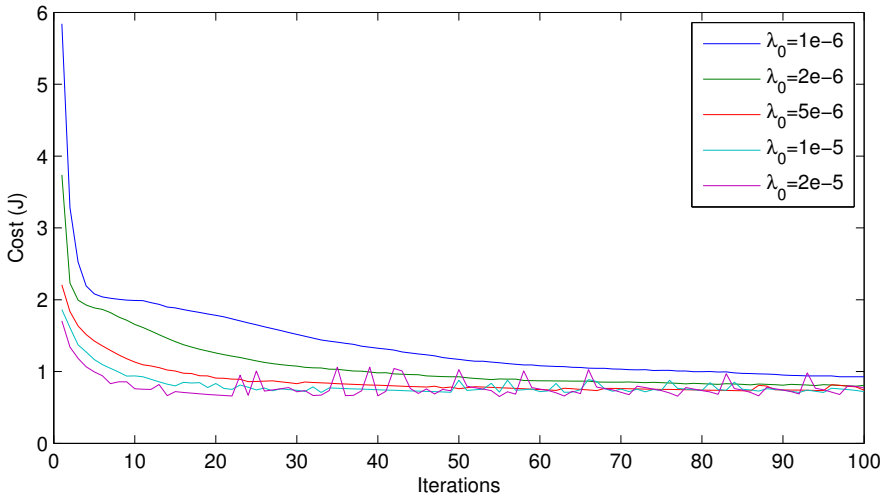

Figure 5.5: Cost of each iteration with varying initial learning rate λ_0. The convergence is faster when increasing λ_0. Although, once λ_0 is large, the solver takes larger steps. It is likely to jump around the minimum. This behavior can be seen from fluctuation in the cost.

Algorithm 5.1 SGD_ML

$y \leftarrow y^0$ {y is an initial set of odometry measurements}
$\lambda \leftarrow \lambda_0$ {Initialize learning rate λ_0}
while y not converged **do**
  select pose i at random
  for each pose i **do**
    $x \leftarrow T(y_i) \otimes .... \otimes T(y_1) \otimes x_0$
    $h \leftarrow map_{ML}(x)$ {defined in eq.(5.24)}
    $g \leftarrow$ gradient around x
    $J \leftarrow$ Jacobian $\partial x_i^h / \partial y$ {eq. (5.30)}
    $d \leftarrow y - y^0$
    $\Delta y \leftarrow \lambda \left(h^{-1} \cdot g \cdot J - (n+1)^{-1} d^T \Sigma^{-1} \right)$ {eq. (5.34)}
    $y \leftarrow y + \Delta y$
  end for
  $\lambda \leftarrow \lambda / (\lambda + 1)$
end while
return y

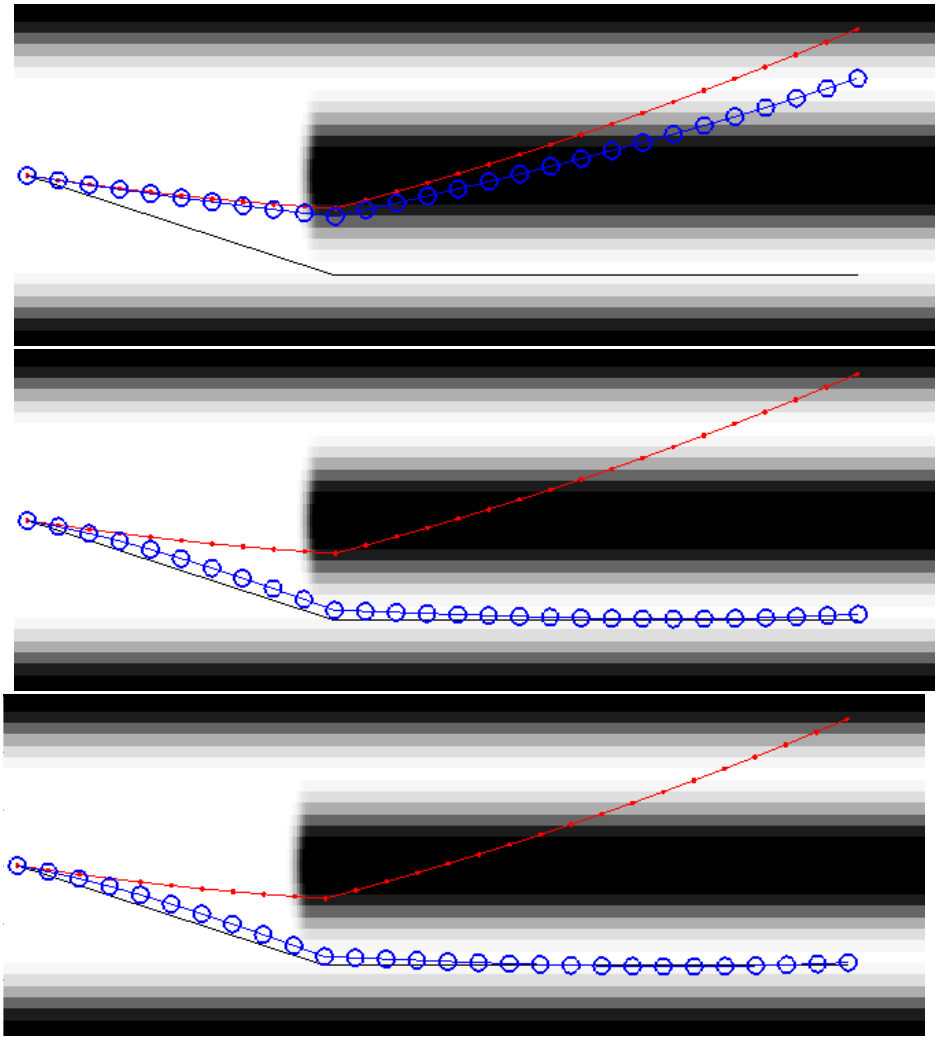

Figure 5.6: A simulation result of the single hypothesis SGD_ML algorithm. The blue line is the maximum likelihood trajectory solution at iteration 1 (top), 2 (middle) and 10 (bottom). The red path is the odometry input from the PDR system. The black line is the ground truth. The solution quickly converges in iteration 2. ($\lambda_0 = 10^{-5}$)

5.6.2 Zippering SGD_ML (SGD_ML_Z)

SGD_ML optimizes the whole trajectory at once. For longer trajectories, this leads to a large search space for the optimizer and may contain several local minima. This problem can be seen in Fig.5.7 when the odometry noise is large, and SGD_ML converges to the wrong solution.

We implement a "zippering" method to further improve robustness to local minima. We start optimizing from the first section of the trajectory and then incrementally move the window forward like a zipper. By beginning optimization in the best-known part of the trajectory, the optimization is more likely to converge to the correct local minima. The process then repeats and moves to the next section. We show that this method is more likely to converge to the correct solution.

As a result, the SGD_ML_Z method improves the robustness of the SGD_ML. A more detailed discussion can be found in the result section.

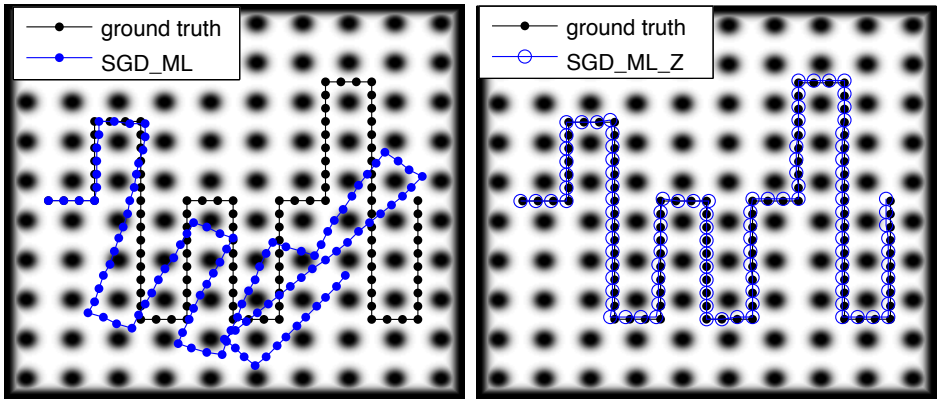

Figure 5.7: (Left) Shows a result of SGD_ML (blue) that converges to a wrong solution when the odometry noise is large. (Right) SGD_ML_Z performs better and converges to the correct solution. The black line is the ground truth.

5.7 Multiple hypothesis tracking

In many situations, the odometry input from the PDR system can lead to ambiguous trajectories. For example,(considering the two identical corridors in Figure 5.8) if the PDR data is uncertain, the human could have taken either path.

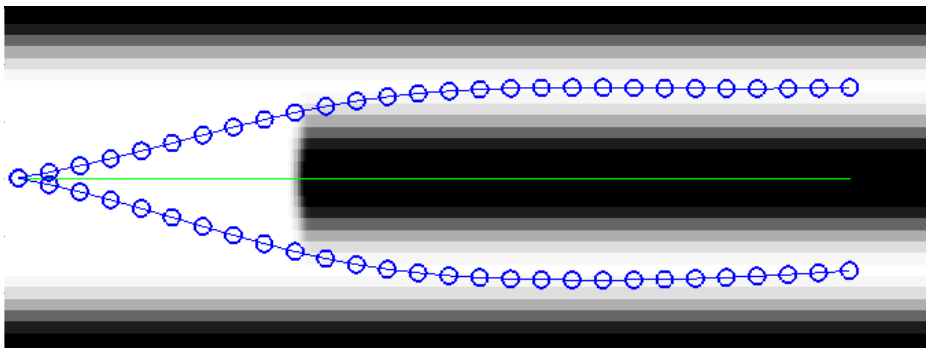

Figure 5.8: The simulated odometry input (green) leads to ambiguous solutions. It leads to right in the middle of two corridors. Both top and bottom trajectories converge into different local minima in the cost function.

For this reason, the robot needs to track multiple hypotheses so it will not commit to the wrong solution when there is a lot of ambiguity. To do this, we will repeat the SGD_ML with different odometry estimates consistent with our odometry noise model, thus exploring other local minima. Specifically, The SGD_ML algorithm is extended by adding two processes, described below.

5.7.1 Perturbation

The initial condition of the state variable y which is normally equal to odometry input y^0 is perturbed to allow the maximum-likelihood solution to converge to different local minima.

We repeatedly sample y from the distribution $N(y^0, \Sigma)$ and perform optimization using SGD_ML. Furthermore, we perform this perturbation when the cost of the current trajectory hypotheses are larger than a threshold indicating poor likelihood. By sampling from the distribution $N(y^0, \Sigma)$, we can make sure that our solutions are

unbiased.

5.7.2 Merging Trajectories

We check that the cost function $J(y)$ between two trajectories (y_1,y_2) is convex or not by using Eq.(5.35). If it is convex (indicating there is a local minimum of the cost function between the two trajectories), they are combined and and kept as a new hypothesis. Two examples of trajectory that can/cannot be merged and their cost as a function of parameter t is shown in Figure 5.9.

$$J(t \cdot y_1 + (1-t) \cdot y_2) \leqslant t \cdot J(y_1) + (1-t) \cdot J(y_2) \qquad \forall t \in [0,1] \tag{5.35}$$

First, the combination of two trajectories is simply the average of both. Finally, the combination is once again optimized by SGD_ML to find the maximum-likelihood solution of the average.

By merging trajectories that are close together, we reduce the number of trajectory hypotheses, thus, reducing the computational complexity. The fine-scale detail of each merged trajectory is lost; however, the route that the leader took is much more interesting for the robot than the fine detail of the paths.

The method is presented in Algorithm 5.2. Figure 5.16 shows the maximum-likelihood solution after both perturbation and merging.

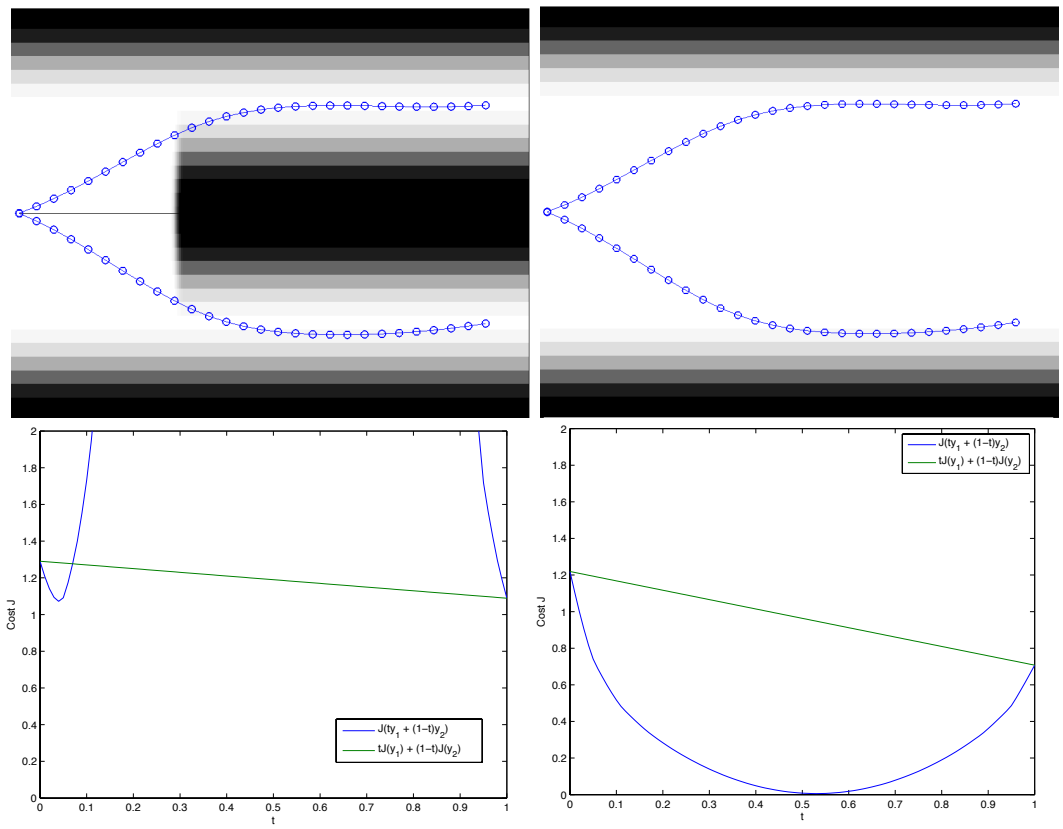

Figure 5.9: (Top)Two trajectories with an obstacle between them (left) and without (right). (Bottom) Cost of the term $J(t \cdot y_1 + (1-t) \cdot y_2)$ and $t \cdot J(y_1) + (1-t) \cdot J(y_2)$ as a fuction of a variable t. On the left the cost function is not convex thus trajectories cannot be merged together while the cost function on the right is convex and can be merged.

Algorithm 5.2 Multiple hypothesis SGD_ML

$n \leftarrow$ number of samples
$s = \{\}$ {s is the sample set}
for loop n times **do**
    sample y from $N(y^0, \Sigma)$
    $y_{ml} \leftarrow SGD\_ML(y)$
    append y_{ml} in s
end for
$s_{merged} \leftarrow merge(s)$
$s_{refined} = \{\}$
for each y in s_{merged} **do**
    $y_{ml} \leftarrow SGD\_ML(y)$
    append y_{ml} in $s_{refined}$
end for
return $s_{refined}$

5.8 Motion between poses

We calculate the cost of a trajectory, recalling from Equation (5.13) that the posterior of a trajectory can be written as:

$$p(x^h|u,z,o) = \eta \prod p(o_i|x^h_{i-1}, x^h_i) \prod p(x^h_i|m_{ML})$$

We can see that the term $\prod p(x^h_i|m_{ML})$ considers only the probability of each pose at x^h_i given a map m_{ML}. However, it does not consider the motion between poses $[x^h_{i-1}, x^h_i]$. This may cause a problem as shown on the left of Figure 5.10. A trajectory passes through a vertical obstacle, which makes it improbable. However, according to the above equation, the probability of this trajectory is instead very likely because the probability of the path is considered only at point x_1 and x_2 and discarded the area in between.

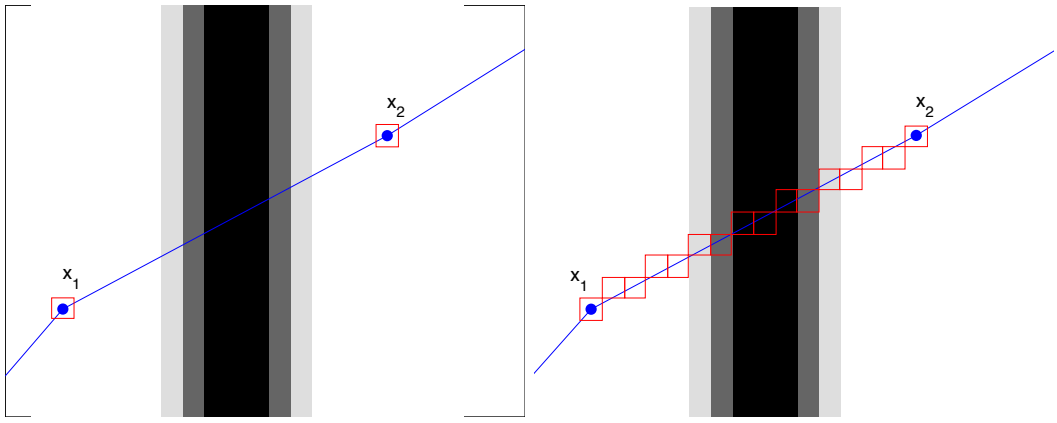

Figure 5.10: A trajectory passes through a vertical obstacle which makes it improbable. (Left) The probability of the path is only considered at each pose; thus it discards the motion between them. (Right) The probability of the path is approximated to be the minimum of all poses along the line, and is thus still able to capture the area in between.

To solve this problem, we assume that the motion between the poses is linear. As a result, the term $p(x^h_i|m_{ML})$ is assumed to be the minimum probability between all cells along the line between x^h_{i-1} and x^h_i. The approximation can capture all the

motion at the poses and between them.

$$p(x_i^h|m_{ML}, x_{i-1}^h) \approx \min p(x|m_{ML}) \qquad x \in line(x_{i-1}^h, x_i^h) \tag{5.36}$$

This above approximation yields better stability than the product of all cells along the line. Since the probability of each cell is always less then one, a longer path that has a higher number of cells in between will have less probability than the shorter ones. This approximation does not affect the optimization process, however, and is used after the optimization step to calculate the actual cost of the trajectory, thus reducing false positive solutions.

5.9 Sample space reduction heuristics

In Section 5.7.1, we apply perturbation steps on a trajectory that has high cost, indicating possible wrong solutions. This may be caused by a trajectory that passes through obstacles or does not agree with odometry inputs.

To find alternative trajectories, our perturbation step in section 5.7.1 samples y from the distribution $N(y^0, \Sigma)$. However, the sample space becomes very large if there are many poses in the trajectory. To reduce the search space, we can start sampling from the last few poses in the trajectory chain in the hope of finding viable solutions quickly. If the sampling does not succeed, we can move further back and sample more poses until a solution is found. This heuristic can find a possible solution much quicker than sampling from the whole trajectory. The results of this heuristic are shown in Figure 5.11.

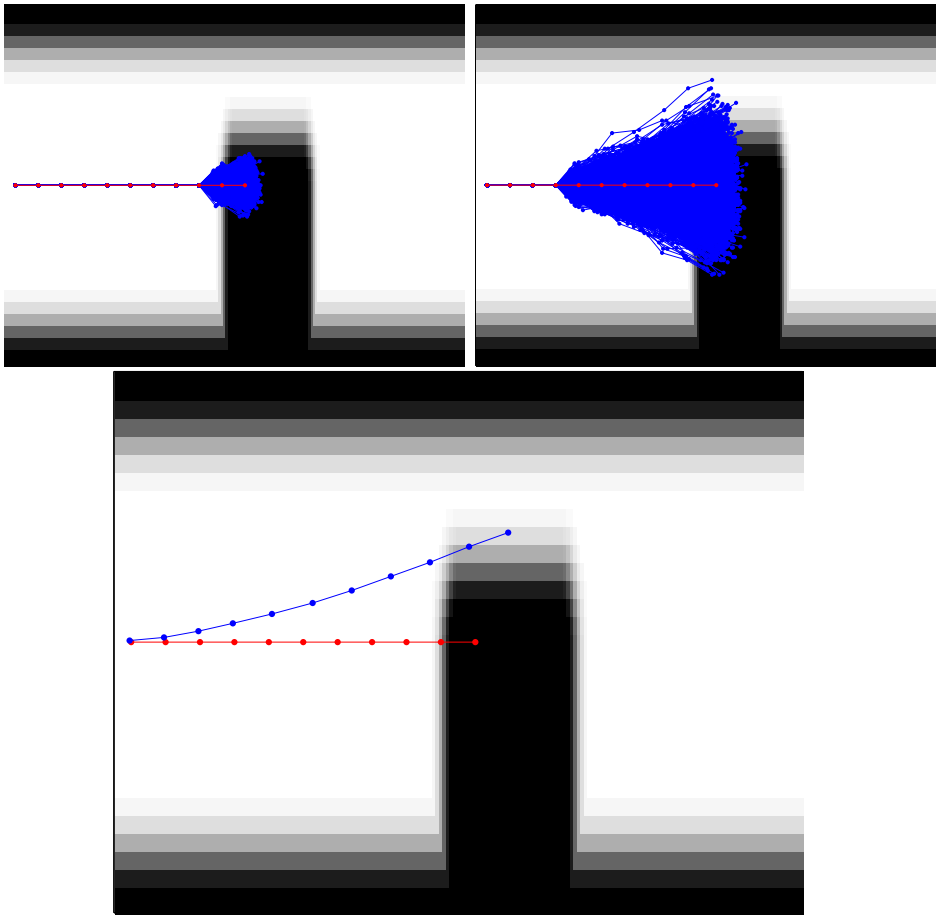

Figure 5.11: (Top-Left) The heuristic starts sampling from the third to last pose (blue) which does not yield any viable solutions. (Top-Right) The sample space moves back 5 more poses and the sample size is large enough to include possible solutions which lead to a new optimal trajectory in the (Bottom) plot, shown in blue.

5.10 Results

We evaluate the performance of both SGD_ML and SGD_ML_Z algorithms by testing them against particle filter tracking as a baseline. First, we repeatedly simulate them in a synthetically generated environment.

5.10.1 Forest world

The forest world is designed to have equally-spaced obstacles ("trees") spread over the map. This obstacle placement has exponentially many possible paths between any two points on the map. The map is also symmetrical in both directions and ambiguous for the tracker. The routes between starting and ending locations are randomly chosen. Furthermore, the simulated odometry measurement is corrupted with noise and fed to all three algorithms.

One particular result is shown in Figure 5.12, in which all three algorithms are comparable. At the same time, however, the SGD_ML algorithm converges to the wrong solution in 25 of 100 runs. An example of this problem is in Figure 5.7. Since the SGD_ML optimizes the whole trajectory at once, this also means that the optimizer tries to optimize a larger search space which may contains several local-minima. As a result, it is possible that the optimization may converge to a different local-minima. Meanwhile, particle filter and SGD_ML_Z do not suffer this issue and converge to the correct solutions in all runs.

Table 5.1 shows the root-mean-square error (RMSE) of the solutions for each algorithm. The RMSE is computed by averaging RMS error of all poses in the trajectory. The SGD_ML algorithm has smaller errors than the particle filter baseline considering only those occasions on which it performs correctly. Compared to the particle filter, it has 35% and 56% improvement in positioning and heading, respectively. Lastly, the SGD_ML_Z performs better than SGD_ML (43% in positioning and 65% in heading) and robustly converges to the correct solutions in all runs. Both improvements over

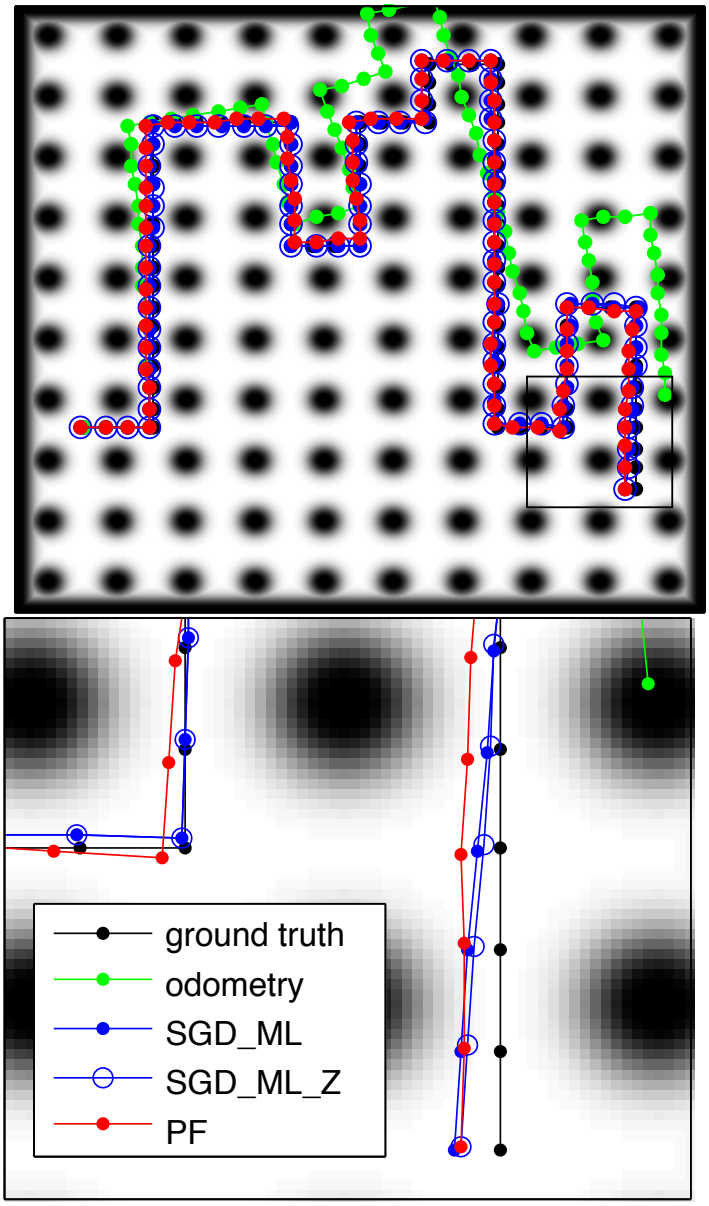

Figure 5.12: (Top) A result from forest world experiment: the obstacles are evenly placed to allow exponentially possible paths between randomly selected start and end locations. The simulated odometry input (green) is corrupted with noise. The black line is the ground truth. The blue and red lines are SGD_ML and particle filter output, respectively. (Bottom) The zoom-in version of the rectangular area in top image.

the particle filter are even larger in larger odometry noise simulation.

On average, our SGD_ML algorithm implemented in MATLAB uses 5.6 times the CPU time of the particle filter baseline. Meanwhile, SGL_ML_Z is 4 times slower than SGD_ML since it optimizes the trajectory repeatedly.

Method	Heading odometry noise σ=0.02 radians			
	Position RMS error (m)		Angular RMS error (radian)	
	Average	Maximum	Average	Maximum
PF	0.2146	0.3436	0.0532	0.0926
SGD_ML*	0.1585	0.2869	0.0340	0.0768
SGD_ML_Z	0.1494	0.2657	0.0322	0.0725

*Note: SGD_ML only converges correctly in 75 of 100 runs.

Method	Heading odometry noise σ=0.05 radians			
	Position RMS error (m)		Angular RMS error (radian)	
	Average	Maximum	Average	Maximum
PF	1.0743	2.5345	0.1472	0.3053
SGD_ML*	0.2470	0.3813	0.0750	0.1348
SGD_ML_Z	0.2312	0.3787	0.0669	0.1382

*Note: SGD_ML only converges correctly in 22 of 100 runs.

Table 5.1: Root-mean-square error (RMSE) comparison between SGD_ML, SGD_ML_Z and particle filter (PF) from 100 forest world runs.

5.10.2 BBB building

We also test the SGD_ML_Z algorithm with real odometry data from the PDR system. The subject walked inside a typical office building. Due to the non-deterministic nature of these algorithms, we repeatedly process each experiment 5 times for better understanding of the performance variation.

From Table 5.3 and Table 5.3, the SGD_ML_Z performs significantly better than the particle filter in both position and angular performance. The SGD_ML_Z has 47% improvement in position and 62% improvement in heading over the particle filter tracking. This result is in agreement with our simulation result in Section 5.10.1. A result from one of the experiments is shown in Figure 5.15.

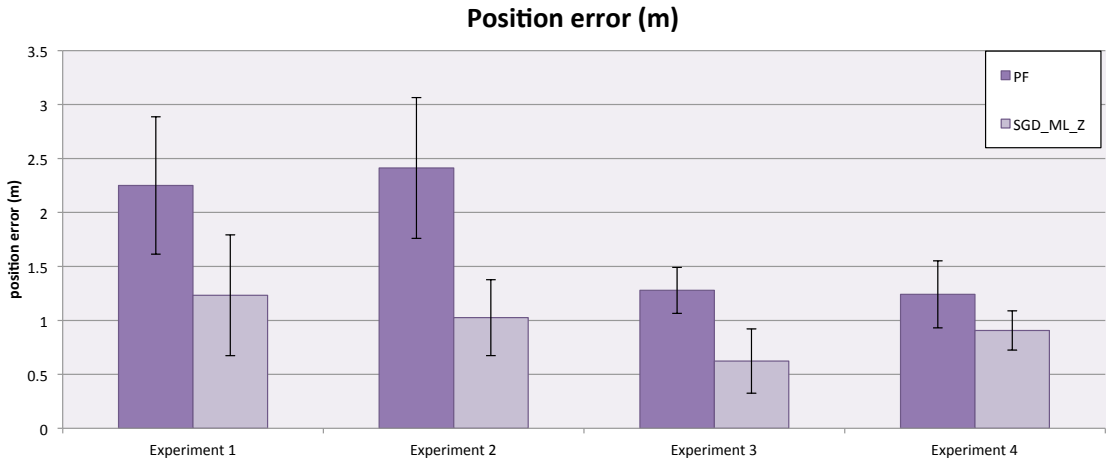

Figure 5.13: Comparision of average position error of BBB building experiments. SGD_ML_Z performs significantly better than particle filter baseline by having lower positioning error.

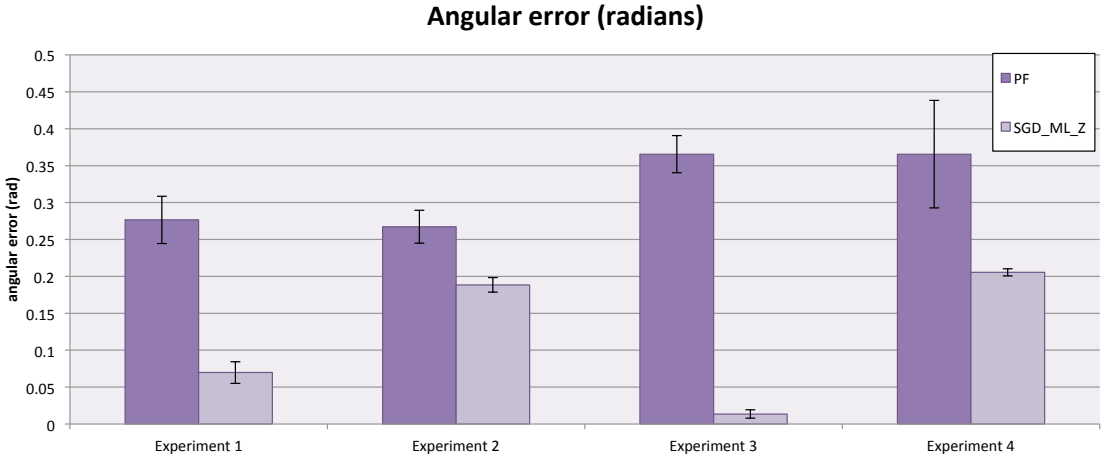

Figure 5.14: Comparision of average angular error of BBB building experiments. SGD_ML_Z performs significantly better than particle filter baseline by having lower angular error.

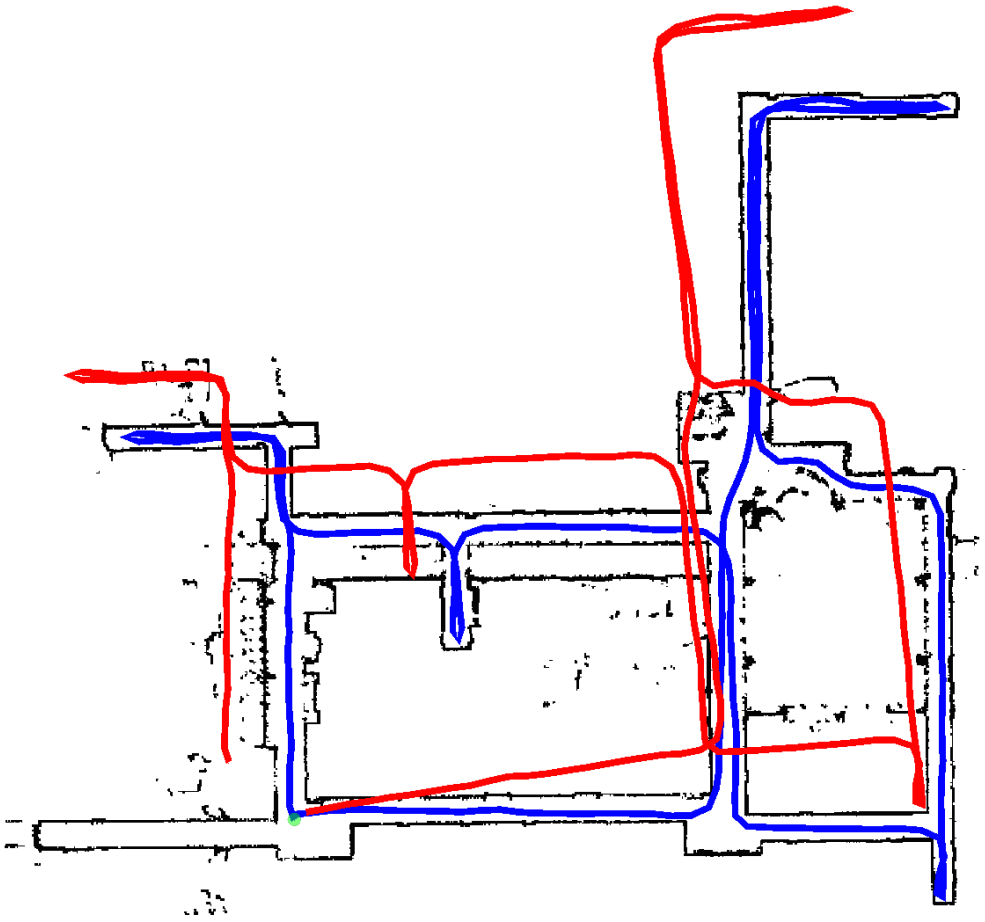

Figure 5.15: An experiment of a subject walked inside a building (Experiment 3 in the table). The map was generated using SLAM from a robot driven around the building. The odometry input recorded from the PDR system is in red and corrupted with noise. The output from SGD_ML_Z algorithm (blue) shows an improvement in position estimation.

	Average position error (m) (95% CI)	
	PF	SGD_ML_Z
Experiment 1	2.250±0.637	1.233±0.5597
Experiment 2	2.412±0.653	1.026±0.3521
Experiment 3	1.278±0.213	0.623±0.2981
Experiment 4	1.241±0.311	0.907±0.1813
Average	1.795±0.610	0.947±0.250

Table 5.2: Average position error of BBB building experiments.

	Average angular error (rad) (95% CI)	
	PF	SGD_ML_Z
Experiment 1	0.277±0.032	0.070±0.0147
Experiment 2	0.267±0.022	0.188±0.0098
Experiment 3	0.365±0.025	0.014±0.0056
Experiment 4	0.365±0.073	0.205±0.0048
Average	0.319±0.053	0.119±0.009

Table 5.3: Average angular error of BBB building experiments.

5.10.3 Multiple hypotheses

The multiple hypotheses version of SGD_ML was also tested in a synthetic map. The map consists of multiple gates that are designed to be ambiguous. The algorithm performs correctly in sampling and merging trajectories to achieve distinct maximum-likelihood solutions. Results from the test can be seen in Fig.5.16.

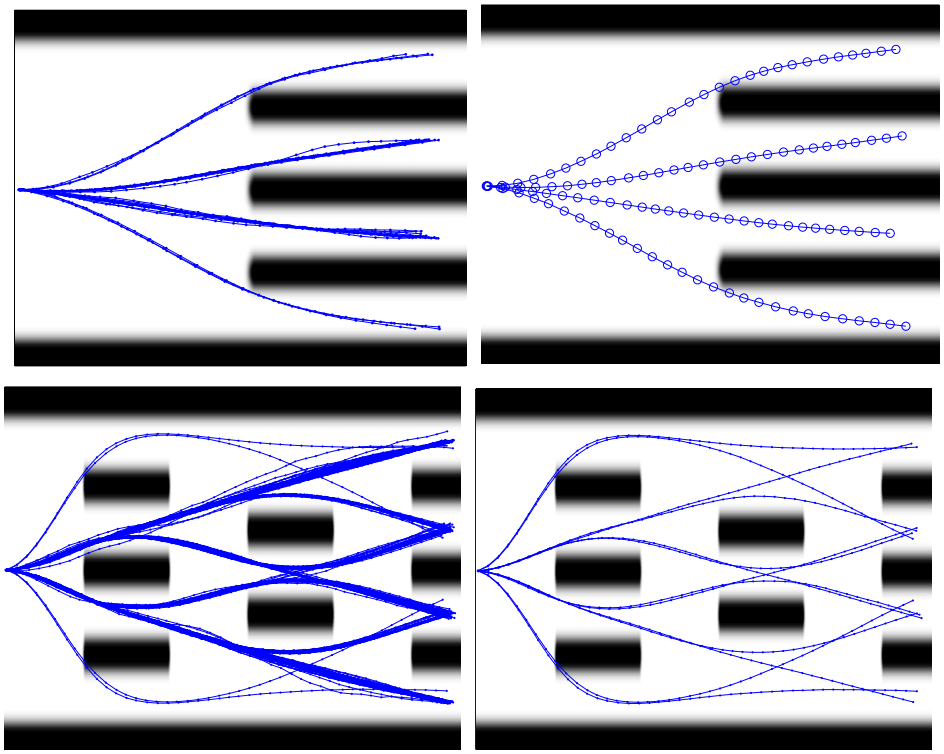

Figure 5.16: (Left) Maximum likelihood hypotheses with perturbed initial condition. (Right) Trajectories after merging. Reducing to 4 and 12 distinct maximum likelihood hypotheses.

5.11 Conclusions

We presented particle filter tracking and three versions of maximum-likelihood tracking using Stochastic Gradient Descent (SGD). The particle filter has the largest margin of error and suffers from particle depletion. The SGD_ML performs well in low noise odometry, but has an increasing probability of failure when the odometry noise is increasing. The SGD_ML_Z achieves better performance than the SGD_ML and robustly performs well in spite of odometry noise. Multiple hypotheses SGD_ML extends the capability of SGD_ML to be able to track multiple hypotheses in case of trajectory ambiguity. Our evaluation characterizes the performance of these three methods and compares them to baseline particle filter tracking.

CHAPTER VI

Conclusion and Future Work

We presented an application of a human leader and a robot follower. Such a system is useful in a "pack mule" application, where the robot carries a heavy load for the human. There is a challenge to this problem is that the robot often lags well behind and out of visual contact with the human.

Instead of relying on Global Positioning System (GPS), which can be jammed or obstructed, our novel approach used a Pedestrian Dead-Reckoning (PDR) system. This approach overcame those limitations by offering a system that is self-contained, immune to GPS interference and does not require the installation of any equipment beforehand.

In addition, we also equipped the robot with high-fidelity mapping sensors and used the observations of the environment together with the PDR trajectory to determine the path taken by the human. One novelty of this method is that it is the first application in which one agent completes the trajectory estimation of another agent while generating its own map of the environment.

The robot has to make a decision of how to make the next movement after the human position has been estimated. The challenging is robot needs to explore the map with the most efficient plan while avoiding obstacles. Though this was not covered in this thesis, this will be the next important topic to work on.

6.1 On the filtering of a Pedestrian Dead-Reckoning (PDR) system

In Chapter III, we found that the original PDR system has large positioning error, especially in elevation. We hypothesized that the original PDR system suffered from an overly-simplified sensor model and a limited state estimation approach.

We analyzed design-choices to improve PDR positioning. Due to the non-linearity of the attitude estimation in the PDR system, we chose the Unscented Kalman Filter (UKF) for state estimation. We performed a comprehensive comparison between model complexities and estimation algorithms including the original PDR system, Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). We presented comparison between a 9, 12 and 15 states of UKF implementation. Furthermore, we compared an UKF system with additive and non-additive noise model.

Our experiments showed that the UKF implementation with 15 states and an additive noise model performed the best. The system improved elevation error on an average of 63% comparing to the original system; however, it was 2.2 times slower.

In the case that we have a rough estimate of the elevations between each floor of a building, we can also construct a heuristic algorithm and feed any prior knowledge of elevation to the UKF to achieve better elevation estimation.

6.2 On the magnetometer-enhanced PDR system in outdoor environment

The positioning error of the PDR system accumulates rapidly without any additional sensors. In Chapter IV, we presented a method to augment the PDR system with a magnetometer to improve positioning in outdoor missions.

Incorporating a magnetometer in a PDR application has a unique challenge. Prior work required the user to perform a calibration procedure, often including rotating

the sensors in full circles or turning the sensor in a specific patterns. Thus, these procedures were not suitable for a sensor mounted on a user's foot. We presented a practical approach that does not require the user to perform any explicit calibration procedure. Our calibration method accounts for mis-alignments between the sensors and the magnetometer's error. We presented a real-time algorithm which can detect magnetic disturbances by comparing the measurement with the earth magnetic model and Inertial Measurement Unit (IMU) measurement.

Our experiments showed that position error of the PDR system with a magnetometer was reduced to 1.9% of the distance travelled. This result was a significant improvement over the PDR system which had 16% of the position error without magnetometer.

6.3 On the maximum-likelihood tracking

In Chapter V, we presented our approach in which the robot combined PDR trajectory received from the human with the robot's own map.

Unlike a conventional localization problem, in this approach one agent completes the trajectory estimation of another agent while generating its own map. We presented several tracking algorithms including a baseline particle filter and maximum-likelihood tracking. Our maximum-likelihood tracking (SGD_ML) utilized Stochastic Gradient Descent (SGD) to estimate the most probable human trajectory. Moreover, we presented enhancements such as zippering technique (SGD_ML_Z) and multiple hypotheses tracking. This resulted in a fast and memory-efficient tracking algorithm with an ability to track multiple possible trajectories.

Our maximum-likelihood tracking outperformed the particle filter in both simulation and real-world experiments. From our real-world experiments, the maximum-likelihood with zippering technique performed the best with an average improvement of 47% in position and 62% in heading over the particle filter tracking.

As for the future work, there is room for refinement for the maximum-likelihood tracking. For example, the map information can be used in the sampling step in the multiple hypotheses SGD_ML algorithm instead of sampling from the whole sample space. This refinement may speed up the computation time of the algorithm.

BIBLIOGRAPHY

BIBLIOGRAPHY

Allan, D. W., and S. Leschiutta (1974), Atomic Frequency Standards, *Alta Frequenza*, *54*(2), 221–230.

Aparicio, C., S. Reader, and R. Cristi (2004), Implementation of a Quaternion-Based Kalman Filter for Human Body Motion Tracking Using MARG Sensors, Ph.D. thesis, Naval Postgraduate School, Monterey, CA.

Ayyappa, E. (1997), Normal human locomotion, part 1: Basic concepts and terminology, *JPO: Journal of Prosthetics and Orthotics*, *9*(1), 10.

Beauregard, S. (2009), Infrastructureless Pedestrian Positioning, Ph.D. thesis, University of Bremen.

Beauregard, S., and H. Haas (2006), Pedestrian dead reckoning: A basis for personal positioning, in *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, pp. 27–35, Shaker Verlag, Hannover, Germany.

Borenstein, J., L. Ojeda, and S. Kwanmuang (2009), Heuristic reduction of gyro drift for personnel tracking systems, *Journal of Navigation*, *62*(1), 41–58.

Cho, S., K. Lee, C. Park, and J. Lee (2003), A personal navigation system using low-cost MEMS/GPS/Fluxgate, in *Proceedings of the 59th Institute of Navigation (ION) Annual Meeting*, pp. 122–127, Institute of Navigation, Albuquerque, NM.

Cobb, S. (1997), GPS pseudolites: Theory, design, and applications, Ph.D. thesis, Stanford University.

Crassidis, J., and F. Markiey (1996), Attitude Estimation Using Modied Rodrigues Parameters, in *Proceedings of the Flight Mechanics/Estimation Theory Symposium*, pp. 71–84, NASA Goddard Space Flight Center, Greenbelt, MD.

Crassidis, J. L., and F. L. Markley (2003), Unscented Filtering for Spacecraft Attitude Estimation, *Journal of Guidance, Control, and Dynamics*, *26*(4), 536–542, doi: 10.2514/2.5102.

El-Sheimy, N., H. Hou, and X. Niu (2008), Analysis and modeling of inertial sensors using allan variance, *IEEE Transactions on Instrumentation and Measurement*, *57*(1), 140–149, doi:10.1109/TIM.2007.908635.

Fajen, B. R., and W. H. Warren (2003), Behavioral dynamics of steering, obstacle avoidance, and route selection, *Journal of experimental psychology. Human perception and performance*, *29*(2), 343–362, doi:10.1167/1.3.184.

Finlay, C. C., et al. (2010), International Geomagnetic Reference Field: The eleventh generation, *Geophysical Journal International*, *183*(3), 1216–1230, doi: 10.1111/j.1365-246X.2010.04804.x.

Fox, D., W. Burgard, H. Kruppa, and S. Thrun (2000), A probabilistic approach to collaborative multi-robot localization, *Autonomous Robots*, *8*(3), 325–344.

Fox, D., S. Thrun, W. Burgard, and F. Dellaert (2001), Particle filters for mobile robot localization, in *Sequential Monte Carlo methods in practice*, pp. 401–428, Citeseer, doi:10.1007/978-1-4757-3437-9_19.

Foxlin, E. (2005), Pedestrian tracking with shoe-mounted inertial sensors, *Computer Graphics and Applications*, *25*(6), 38–46.

Gallier, J. (2011), The Quaternions and the Spaces S3, SU(2), SO(3), and RP3, in *Geometric Methods and Applications*, *Texts in Applied Mathematics*, vol. 38, pp. 281–300, Springer New York, doi:10.1007/978-1-4419-9961-0_9.

García, R. (2012), Hop! The following suitcase, http://cargocollective.com/ideactionary/hop.

Gebre-Egziabher, D., G. Elkaim, J. Powell, and B. Parkinson (2001), A non-linear, two-step estimation algorithm for calibrating solid-state strapdown magnetometers, in *8th International Conference on Integrated Navigation Systems*, pp. 290–297, St. Petersburg, Russia.

Guennebaud, G., and B. Jacob (2010), Eigen V3: C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms, http://eigen.tuxfamily.org.

Hofmann-Wellenhof, B., H. Lichtenegger, and J. Collins (1994), *Global Positioning System: Theory and Practice*, 3rd ed., Springer-Verlag, New York.

Huber, P. J. (2011), *Robust statistics*, Springer, New York.

Jimenez, A. R., F. Seco, C. Prieto, and J. Guevara (2009), A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU, in *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pp. 37–42, IEEE, Budapest, Hungary.

Judd, T. (1997), A personal dead reckoning module, in *ION GPS*, vol. 9, pp. 47–51, Kansas City, MO.

Julier, S., J. Uhlmann, and H. Durrant-Whyte (1995), A new approach for filtering nonlinear systems, in *Proceedings of 1995 American Control Conference - ACC'95*, vol. 3, pp. 1628–1632, IEEE, Seattle, WA, doi:10.1109/ACC.1995.529783.

Kwanmuang, S., J. Borenstein, and L. Ojeda (2010), Personal Dead-Reckoning System for GPS-denied Outdoor Environments, *Tech. rep.*

Kwanmuang, S., L. Ojeda, and J. Borenstein (2011), Magnetometer-enhanced personal locator for tunnels and GPS-denied outdoor environments, in *Proceedings of the SPIE Defense, Security + Sensing; Unmanned Systems Technology XIII*, International Society for Optics and Photonics, Orlando, FL, doi:10.1117/12.885346.

Lefferts, E., F. Markley, and M. Shuster (1982), Kalman Filtering for Spacecraft Attitude Estimation, *Journal of Guidance, Control, and Dynamics*, *5*(5), 417–429, doi:10.2514/3.56190.

Markley, F. L. (1978), Parameterization of the Attitude, in *Spacecraft Attitude Determination and Control*, chap. 12, pp. 410–421, Springer Science & Business Media.

Markley, F. L. (2003), Attitude Error Representations for Kalman Filtering, *Journal of Guidance, Control, and Dynamics*, *26*(2), 311–317, doi:10.2514/2.5048.

Maus, S., S. Macmillan, S. McLean, B. Hamilton, A. Thomson, M. Nair, and C. Rollins (2010), The US/UK World Magnetic Model for 2010-2015, *Tech. rep.*, U.S. National Oceanographic and Atmospheric Administration, doi: 10.7289/V5TH8JNW.

Mezentsev, O., J. Collin, and G. Lachapelle (2005), Pedestrian Dead Reckoning–A Solution to Navigation in GPS Signal Degraded Areas?, *Geomatica*, *59*(2), 175–182.

Naroditsky, O., Z. Zhu, A. Das, S. Samarasekera, T. Oskiper, and R. Kumar (2009), Videotrek: A vision system for a tag-along robot, in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pp. 1101–1108, IEEE, Miami Beach, FL, doi: 10.1109/CVPRW.2009.5206696.

Ojeda, L., and J. Borenstein (2007a), Non-GPS navigation for security personnel and first responders, *Journal of Navigation*, *60*(3), 391–407, doi: 10.1017/S0373463307004286.

Ojeda, L., and J. Borenstein (2007b), Personal Dead-reckoning System for GPS-denied Environments, in *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pp. 1–6, IEEE, Rome, Italy, doi: 10.1109/SSRR.2007.4381271.

Olson, E., J. Leonard, and S. Teller (2006), Fast iterative alignment of pose graphs with poor initial estimates, in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 2262–2269, IEEE, Orlando, FL, doi: 10.1109/ROBOT.2006.1642040.

Petkov, P., and T. Slavov (2010), Stochastic modeling of MEMS inertial sensors, *Cybernetics and information technologies*, *10*(2), 31–40.

Randell, C., C. Djiallis, and H. Muller (2003), Personal position measurement using dead reckoning, in *Proceedings of the Seventh IEEE International Symposium on Wearable Computers (ISWC03)*, pp. 166–173, IEEE, White Plains, NY, doi: 10.1109/ISWC.2003.1241408.

Robbins, H., and S. Monro (1951), A Stochastic Approximation Method, *The Annals of Mathematical Statistics*, *22*(3), 400–407, doi:10.1214/aoms/1177729586.

Schaub, H., and J. L. Junkins (1996), Stereographic orientation parameters for attitude dynamics: a generalization of the Rodrigues parameters, *Journal of the Astronautical Sciences*, *44*(1), 1–19.

Teck Chew, N., J. Ibanez-Guzman, S. Jian, G. Zhiming, W. Han, and C. Chen (2004), Vehicle following with obstacle avoidance capabilities in natural environments, in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, pp. 4283–4288, IEEE, New Orleans, LA.

Titterton, D., and J. L. Weston (2004), *Strapdown inertial navigation technology*, vol. 17, 2 ed., IET.

Trawny, N., and S. I. Roumeliotis (2005), Indirect Kalman Filter for 3D Attitude Estimation, *Tech. rep.*, Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN.

Trefethen, L. N., and D. Bau (1997), *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics.

Van der Merwe, R., N. de Freitas, A. Doucet, and E. Wan (2001), The unscented particle filter, in *Kalman filtering and neural networks*, pp. 221–280, Wiley, New York, doi:CUED/F-INFENG/TR380.

Weinberg, H. (2002), Using the ADXL202 in pedometer and personal navigation applications, http://application-notes.digchip.com/013/13-14984.pdf.

Woodman, O. (2010), Pedestrian localisation for indoor environments, Ph.D. thesis, University of Cambridge, doi:10.1145/1409635.1409651.

Yun, X., and E. R. Bachmann (2006), Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking, *IEEE Transactions on Robotics*, *22*(6), 1216–1227.

Zampella, F., M. Khider, P. Robertson, and A. Jiménez (2012), Unscented Kalman filter and Magnetic Angular Rate Update (MARU) for an improved Pedestrian Dead-Reckoning, in *IEEE PLANS, Position Location and Navigation Symposium*, pp. 129–139, IEEE, Myrtle Beach, SC, doi:10.1109/PLANS.2012.6236874.