Enhanced Stance Phase Detection and Extended Kalman Filtering for Strapdown Pedestrian Dead Reckoning

MSc in Mobile and Ubiquitous Computing

Poorna Talkad Sukumar

September 2010

ABSTRACT

Pedestrian Dead Reckoning (PDR) using shoe-mounted inertial sensors is a self-contained navigation technique which could address the needs of emergency responders. An improved PDR system has been implemented in this thesis using an extended Kalman filter (EKF) formulation with periodic zero-velocity updates (ZUPTs). The system has been evaluated for a variety of data sets recorded using the Xsens MTx inertial measurement unit. The PDR system was implemented both with and without using the Xsens's orientation to also determine if we need to use the expensive Xsens device with its 'black box' algorithm or if we can use less expensive inertial sensors with our own algorithms. Different stance phase detection methods have also been implemented in this thesis.

Contents

List of Figures

Chapter 1

Introduction

Location and navigation support is essential in emergency response scenarios. GPS is not available indoors and all the current indoor navigation systems require installation of some infrastructure or an instrumented, marked or pre-mapped environment. Hence, they are impractical for use by emergency responders such as firefighters. Emergency responders must be able to keep track of their locations and reach safety quickly under dangerous conditions such as smoke and power failure [1–4].

Therefore, there is a need for a self-contained navigation system which will work reliably in any indoor or outdoor setting. Pedestrian Dead Reckoning (PDR) using shoe-mounted inertial sensors could address the needs of emergency responders. Dead reckoning is a fundamental method which consists of adding a moving body's measured or calculated change in position to its previous position to obtain its current position. Inertial sensors usually comprise of three accelerometers and three gyroscopes measuring the accelerations and angular rates in the x, y and z directions respectively. These inertial sensor outputs are integrated to obtain the velocity, position and attitude which are used to determine the current position with respect to a known reference using the dead reckoning method.

But unaided inertial navigation systems suffer from drift. The small errors in the ac-

celeration are integrated into increasingly larger errors in the velocity which are integrated into even larger errors in position. Therefore, the position has to be periodically corrected using techniques such as zero velocity updates (ZUPTs). In pedestrian navigation, the foot alternates between a moving swing phase and a stationary stance phase. Zero velocity updates (ZUPTs) are applied during the stance phases since the velocity during those periods should theoretically be zero. ZUPTs limit the growth of the velocity errors in the inertial navigation system.

Previous work on PDR include using shoe-mounted inertial sensors and applying periodic ZUPTs [1, 3–8]. An extended Kalman filter (EKF) is also used in some papers to combine the inertial navigation velocity solution with the zero velocity measurements [1,5,6,8]. Using an EKF not only helps in estimating the velocity errors, but the position and attitude errors, and accelerometer and gyro biases are also estimated.

But most of the previous papers mentioned above have been able to achieve good tracking only for a few minutes. Also, the inertial sensors used and the environments in which the traces were collected differ among the different implementations.

This thesis extends previous work to build a more robust and accurate dead reckoning system to achieve good tracking for a variety of data sets each having a trace approximately a few minutes in duration. These data sets were already available and were all recorded using the Xsens MTx inertial measurement unit [1] firmly attached to the shoe as shown in figure 1.1. The Xsens inertial measurement unit (IMU) has an in-built real-time algorithm which fuses the sensor outputs to calculate the 3D orientation. EKF/ZUPTs formulation has been used to implement the PDR system both with and without using the Xsens's orientation to also determine if we need to use the expensive Xsens device with its 'black box' algorithm or if we can use less expensive inertial sensors with our own algorithms. Different stance phase

[1] Xsens website. http://www.xsens.com/

Figure 1.1 Xsens MTx IMU firmly attached to the shoe

detection methods have also been implemented in this thesis. All the implementations have been developed using *MATLAB*.

The forthcoming chapters of this thesis are structured as follows; 'Background and Experimental Setup' briefly discusses related work in the PDR field with formal introductions to the extended Kalman filter (EKF) and strapdown inertial navigation system (INS). A summary of the different data sets used in this thesis is also given; 'Stance Phase Detection and Basic EKF/ZUPTs Pedestrian Tracking' describes the three stance phase detection algorithms implemented in this thesis and explains the PDR system with EKF/ZUPTs formulation using the Xsens's orientation; 'EKF tracking using raw IMU measurements' describes the PDR system implementation where orientation is calculated using raw IMU measurements (accelerometer and gyroscope data) and additional methods for reducing the heading drift are also discussed, and finally the 'Conclusions' highlights the key contributions and important observations of this thesis and topics for future work are also listed.

Chapter 2

Background and Experimental Setup

A summary of the related work on pedestrian dead reckoning using shoe-mounted inertial sensors is given in section 2.1. Formal introductions to strapdown inertial navigation systems and the extended Kalman filter are given in sections 2.2 and 2.3, respectively. The various data sets used in the thesis are summarized in section 2.4.

2.1 Related Work

A PDR system with shoe-mounted inertial sensors using periodic application of zero-velocity updates (ZUPTs) has been implemented by both Beauregard [4], and Ojeda and Borenstein [3]. In the implementation by Beauregard [4], the results obtained are fairly accurate in terms of distance travelled and handling arbitrary manoeuvres including side/back stepping, criss-cross walking and on the spot turns which are typical of first responders' walking patterns. The system implemented by Ojeda and Borenstein [3] performs very well for different gaits as well as on stairs, slopes and generally on three dimensional terrain.

One of the important developments in the paper by Feliz et al [7] is the filter developed for stance phase detection using which it can be determined if the foot is on the ground or in

motion with higher accuracy. The exact start and end of the stance phase can be determined and zero-velocity updates are applied during the entire stance phases detected.

In the paper by Faulkner et al [5], the fundamental issues affecting the altitude accuracy in systems with boot mounted IMU and co-located magnetic compass and using EKF/ZUPTs formulation, has been described. It has been found that the main causes affecting the altitude accuracy are the inadequate dynamic range of the accelerometers and gyros to fully sense the boot's motion and inadequate detection of ZUPT periods.

EKF/ZUPTs formulation is used by Foxlin [1] for the PDR system with shoe-mounted IMU, to achieve additional benefits. The EKF not only estimates the velocity errors but the position and attitude errors, and accelerometer and gyro biases are also estimated.

The basic PDR implementations by Jiménez et al [6] and Sujatha [8] are based on Foxlin's implementation [1]. But, additional methods for reducing attitude and heading errors are also described.

It should be noted that the inertial sensors used differ among the different implementations.

2.2 Strapdown Inertial Navigation System

'Navigation' is the determination of the position and velocity of a moving body with respect to a known reference. Many navigation techniques are based on the fundamental method of 'dead reckoning'. Dead reckoning either measures the change in position or measures the velocity and integrates it to obtain the change in position. This is added to the previous position to obtain the current position of the moving body. Since the velocity and the change in position are measured in the body coordinate frame, a separate attitude measurement is required to determine the direction of travel in the global (or navigation) frame [9].

An inertial navigation system (INS) is a complete three-dimensional dead-reckoning nav-

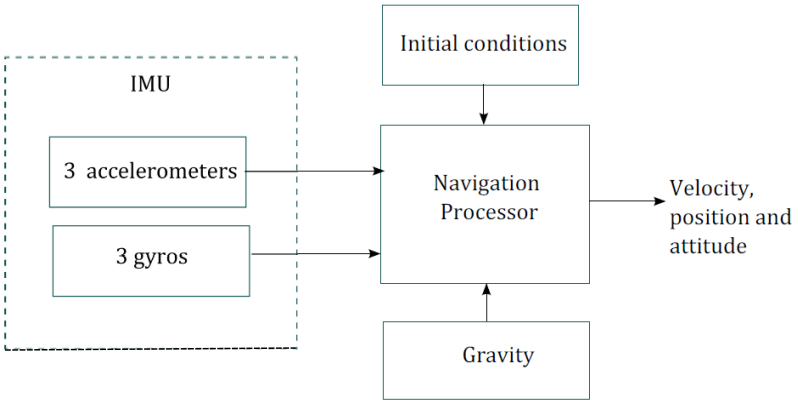

Figure 2.1 Basic principle of an inertial navigation system (adapted from [9]).

igation system. It comprises of an inertial measurement unit (IMU) and a navigation processor as shown in figure 2.1. The IMU consists of three accelerometers and three gyroscopes which measure the accelerations and angular rates in the three mutually orthogonal directions x, y and z respectively. The navigation processor integrates the sensor outputs to give the navigation solution (velocity, position and the attitude) [9].

In strapdown inertial navigation systems, the inertial sensors are strapped to the object and therefore, the sensor outputs are measured in the sensor (or body) frame rather than the global (or navigation or reference) frame. The attitude (or orientation) is calculated by integrating the angular rates measured by the three gyros. This orientation is then used to resolve the accelerations measured in the sensor frame into the global frame and the gravitational component is removed from the z-acceleration. The global accelerations are then integrated to obtain the velocity and the velocity is integrated to obtain the position [10].

The basic principle of a navigation processor is shown in figure 2.2. Before an INS is used to provide a navigation solution, that navigation solution must be initialized. Position and velocity must be initialized using external information but attitude can be initialized either by using external information or by sensing gravity and the Earth's rotation [9].

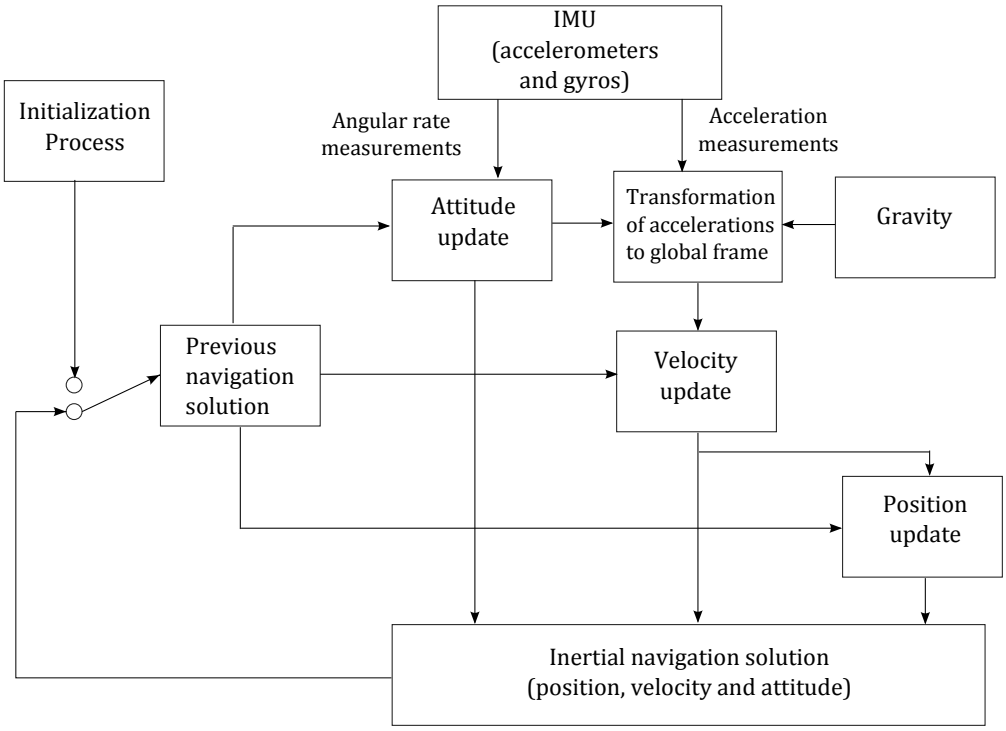

Figure 2.2 Basic principle of an inertial navigation processor (adapted from [9]).

Inertial navigation is used in a wide range of applications including aircraft navigation, tactical and strategic missiles, spacecraft, submarines, ships and pedestrian navigation. Recent advances in the manufacturing of micro-electro-mechanical (MEMS) devices have made it possible to manufacture small and light inertial navigation systems [10].

2.3 The Extended Kalman Filter

The Kalman filter is an estimation algorithm used in many applications, particularly in the area of autonomous or assisted navigation. Its uses include maintaining a smoothed navigation solution in satellite navigation systems, alignment and calibration of inertial navigation systems, and combining the various outputs in integrated navigation systems [9].

It estimates a number of parameters that define the state of a system, such as position

and velocity, that may continuously change, in real-time. The estimates are updated using measurements which are functions of the parameters estimated and the measurements are generally corrupted with noise. At a given time, the measurements may not contain sufficient information to uniquely determine the values of the parameters at the time.

The Kalman Filter is a recursive estimator and has two distinct phases, 'time update' and 'measurement update'. During the time update phase, the state estimate from the previous time step is used to predict an estimate of the state at the current time step. This predicted state estimate is also known as the *a priori* state estimate because it does not include the measurement from the current time step. The Kalman filter also maintains a measure of the uncertainties in its estimates and the correlations between the errors in the estimates of the different parameters. During the measurement update phase, the *a posteriori* estimate is obtained by computing a weighted average of the *a priori* estimate and the measurements from the current time step. Most weight is given to the value with the least uncertainty.

The key element in the recursive procedure is the use of the results of the previous step to obtain the desired result for the current step. There is no need to store all the previous measurements and the previous computational effort is used to good advantage [11].

In a standard Kalman filter, the system and the measurement models are assumed to be linear. This is not always the case for real systems. For situations with nonlinear dynamics and/or nonlinear measurement relationships, an extended Kalman filter (EKF) can be used which linearizes about a trajectory that is continuously updated with the state estimates resulting from the measurements as shown in figure 2.3.

The time update and measurement update equations for an EKF are given in tables 2.1 and 2.2, respectively [12]:
where,

1. $\hat{x}_{k+1|k}$ and $\hat{x}_{k+1|k+1}$ are the *a priori* and *a posteriori* estimates of the state at time step

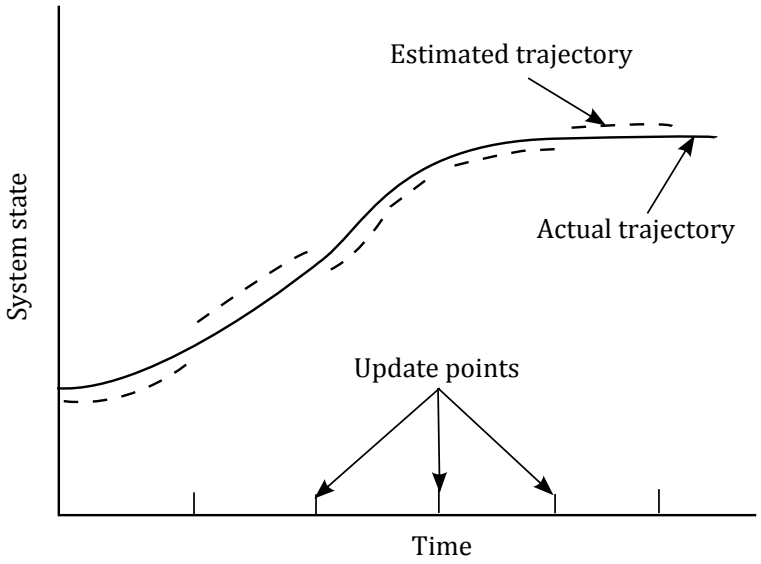

Figure 2.3 Estimated and actual trajectories for an extended Kalman filter (adapted from [11]).

EKF time update equations
$\hat{x}_{k+1\|k} = f(\hat{x}_{k\|k})$
$P_{k+1\|k} = F_{k+1} P_{k\|k} F_{k+1}^T + Q$

Table 2.1 EKF time update equations

$k+1$, respectively.

2. f is the non-linear function that relates the state at previous time step k (x_k), to the state at the current time step $k+1$ (x_{k+1}) and h is the non-linear function that relates the state at time step $k+1$ (x_{k+1}) to the measurement at time step $k+1$ (z_{k+1}).

3. $P_{k+1|k}$ is the error covariance matrix at time $k+1$ based on measurements received through k and $P_{k+1|k+1}$ is the updated error covariance matrix at time $k+1$ using the measurement received at $k+1$.

4. Q and R are the system noise and measurement noise covariance matrices.

EKF measurement update equations
$K_{k+1} = P_{k+1\|k} H_{k+1}^T (H_{k+1} P_{k+1\|k} H_{k+1}^T + R)^{-1}$
$\hat{x}_{k+1\|k+1} = \hat{x}_{k+1\|k} + K_{k+1}(z_{k+1} - h(\hat{x}_{k+1\|k}))$
$P_{k+1\|k+1} = (I - K_{k+1} H_{k+1}) P_{k+1\|k}$

Table 2.2 EKF measurement update equations

5. F_{k+1} and H_{k+1} are the system and measurement, respectively. F_{k+1} is the Jacobian matrix of f with respect to state x and H_{k+1} is the Jacobian matrix of h with respect to x.

6. K_{k+1} is the Kalman gain at time step $k+1$.

It is common in inertial navigation systems to use a complementary Kalman filter which estimates the errors in the state variables rather than the states variables (section 3.2.2).

2.4 Data sets used in this thesis

A summary of the different data sets used in this thesis is given below.

2.4.1 Long corridor

These data sets were recorded on a floor at Lancaster University's InfoLab21. The path travelled, starting at (0,0) and consisting of three straight lines, is shown in figure 2.4 along with the actual distances travelled (4.79m, 45.56m and 34.45m along the three straight lines from the starting point, respectively). Three different users walked once along the path, took a turn and returned to the starting point along the same path covering a total distance of 169.6m. These data sets (L1, L2, L3) have a duration of approximately 2.5 minutes

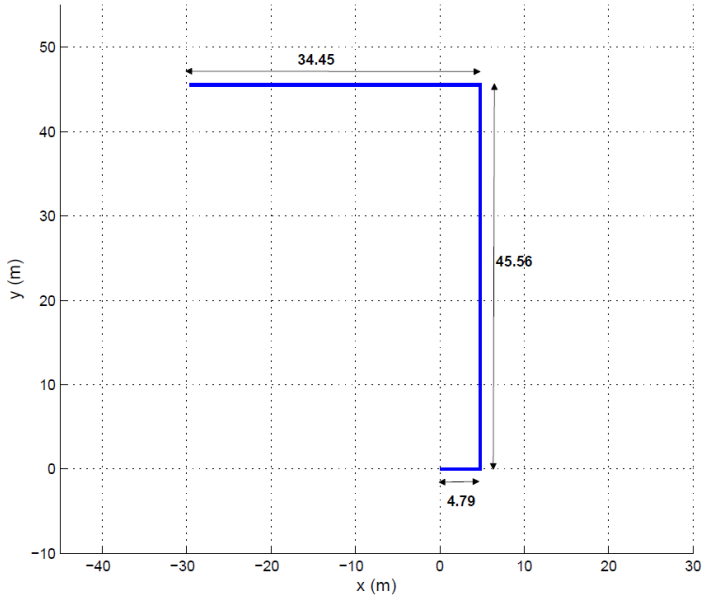

Figure 2.4 Path travelled along corridors in InfoLab21

each. One of the users repeated the path a number of times (6 times) and returned to the starting point covering a total distance of 508.8m and this data set (L4) has a duration of approximately 5.8 minutes.

2.4.2 Data collected in University of Twente

Data was recorded in University of Twente for different trail topologies, durations, walking speeds and weighting factor set for the Xsens MTx sensor output [13]. The weighting factor indicates how much the magnetometer data should be weighted relative to the accelerometer data in the calculation of orientation [14]. For example, in cases of magnetic disturbances, less weight is assigned to the magnetometers. The experiments were conducted in and around multiple rooms and a corridor connecting them. The user travelled the different trajectories at least five times and occasionally stopped at some predefined points. Real-

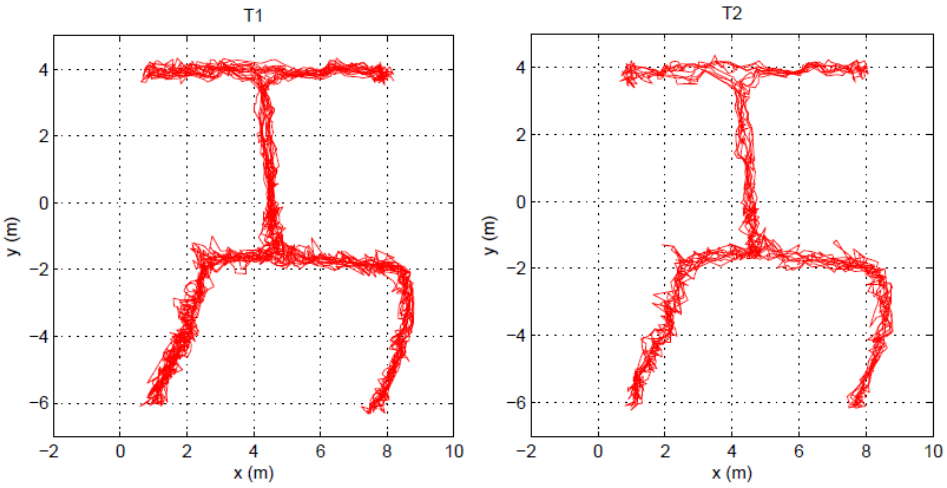

Figure 2.5 Ubisense plots for T1 and T2

time tracking was also recorded for the above trajectories using Ubisense.[1] The data sets used in this thesis are described below and their corresponding Ubisense plots are shown in the figures 2.5 and 2.6.

The Twente data sets are:

(1) T1 and (2) T2 (figure 2.5): These data sets consist of a straight path walk between multiple rooms along either side of the corridor. The approximate durations are 9 minutes for (1) and 5.5 minutes for (2). Magnetometer data is not used for the orientation computation in (1) and accelerometer data is given more weight relative to magnetometer data in the orientation computation in (2).

(3) T3 and (4) T4 (figure 2.6): These data sets consist of a walk along the boundary of the rooms. (3) was recorded for a slow walking pattern and (4) was recorded for a fast walking pattern and they are of approximately 6 and 3 minutes in duration, respectively. Accelerometer data is given more weight relative to magnetometer data in the orientation computation in both (3) and (4).

[1]Ubisense Technologies. http://www.ubisense.net/

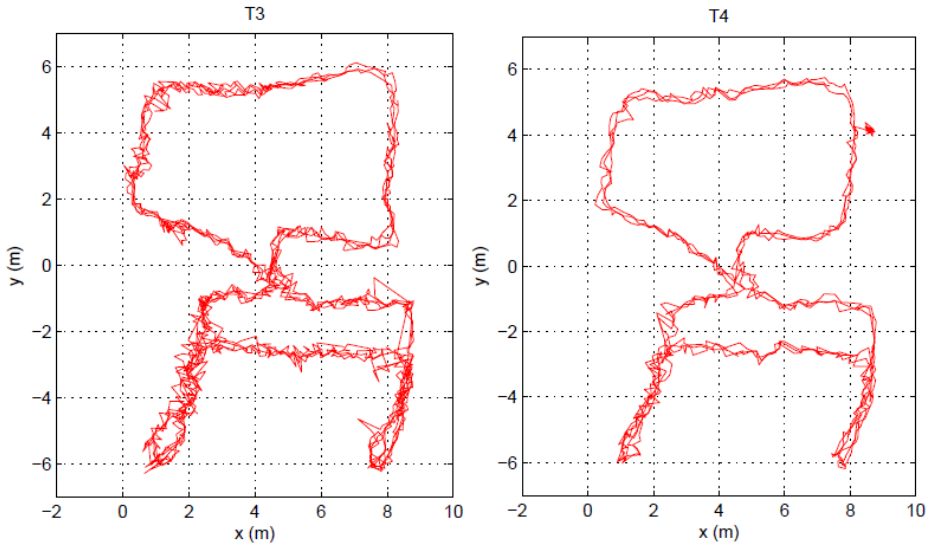

Figure 2.6 Ubisense plots for T3 and T4

2.4.3 Biba Workshop

A user walked a complex path, including two spiral staircases, covering over 200m in a large industrial workshop with heavy machinery (magnetic interference). The path travelled is shown on the Biba workshop floor plan in figure 2.7. This data set has a duration of approximately 4 minutes.

Slightly older versions of the Xsens MTx IMU were used to record the data in (ii) and (iii). Only the in-built software for orientation computation is supposedly better in the latest Xsens MTx IMU (used in (i)) than those used in (ii) and (iii), but no great changes were observed in the results between these versions. Moreover, other factors such as different sensor output settings or level of magnetic interference in the buildings also affect the results. This is highlighted in the analysis, where appropriate.

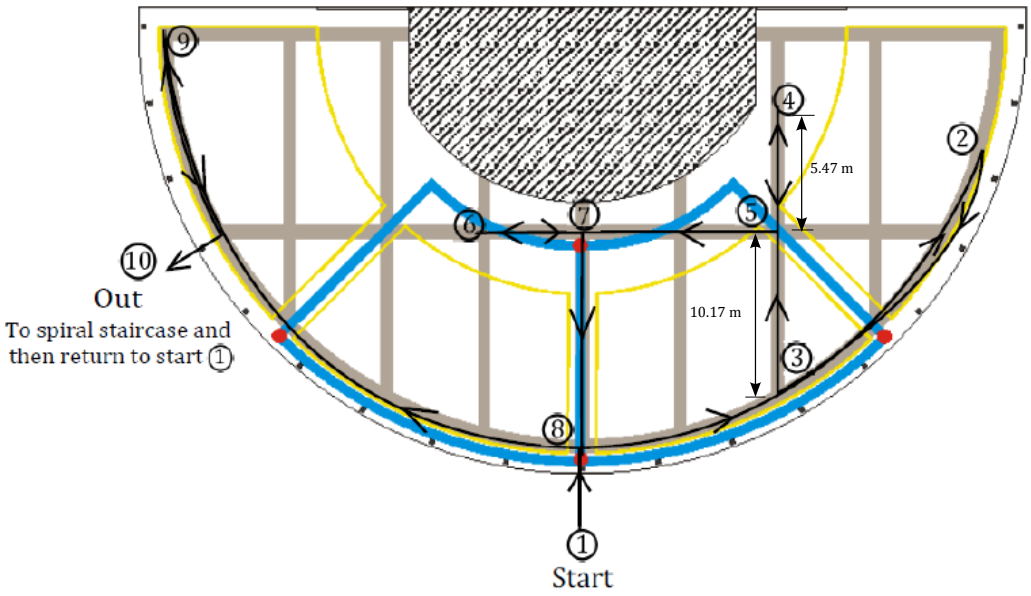

Figure 2.7 Path travelled in Biba workshop

Chapter 3

Stance Phase Detection and Basic EKF/ZUPTs Pedestrian Tracking using Xsens orientation

This chapter describes the stance phase detection algorithms implemented in this thesis and the basic EKF/ZUPTs formulation. Section 3.1 gives formal descriptions of zero-velocity updates (ZUPTs) and benefits of using an extended Kalman filter (EKF). Section 3.2 describes the EKF/ZUPTs formulation using Xsens orientation. The three stance phase detection algorithms implemented in this thesis are explained in section 3.3. Results of the various implementations described in sections 3.2 and 3.3 are presented in sections 3.4 and 3.5. The values used for the EKF parameters for all the implementations in this chapter are given in the 'Filter tuning' section (section 3.6) and a summary of the results is given in section 3.7.

3.1 ZUPTs and Benefits of using EKF

This section describes the zero-velocity updates (ZUPTs) technique, the general advantages of using a Kalman filter in integrated navigation systems and the additional benefits of using an extended Kalman filter for applying ZUPTs in a pedestrian navigation system.

Unaided inertial navigation systems suffer from drift: small errors in the acceleration and angular rate measurements are integrated into increasingly larger errors in the velocity which are integrated into even greater errors in the position. Therefore the position must be periodically corrected using information provided by one or more navigation aids such as external measurements from GPS or techniques like zero-velocity updates (ZUPTs).[1]

ZUPTs are particularly useful for pedestrian navigation with a shoe-mounted IMU. During normal walking, a person's foot alternates between a stationary stance phase and a moving stride phase. The ZUPTs are applied during the stationary stance phase when the foot comes in contact with the ground because the velocity during that period, which lasts for about 0.4 seconds, should theoretically be zero. Therefore, performing ZUPTs on every step helps in replacing the cubic-in-time error growth with one that is linear in the number of steps [1–7, 9, 15]. A plot of the 2D velocity of a shoe-mounted IMU with and without ZUPTs, represented by dotted black and solid red lines, respectively, is shown in figure 3.1.

In integrated navigation systems, where INS outputs are combined with information provided by one or more navigation aids, a filtering technique is used to combine all the available measurements, which are generally corrupted with noise, to provide a better estimate than could be obtained using any single measurement. Kalman filtering is one of the filtering techniques used for the fusion of data in such systems. The difference between the INS estimates and the reference are input to a Kalman filter which corrects the velocity, attitude, and sometimes the position. It also often estimates the INS instrument errors such as

[1] Inertial Navigation System. http://en.wikipedia.org/wiki/Inertial_navigation_system

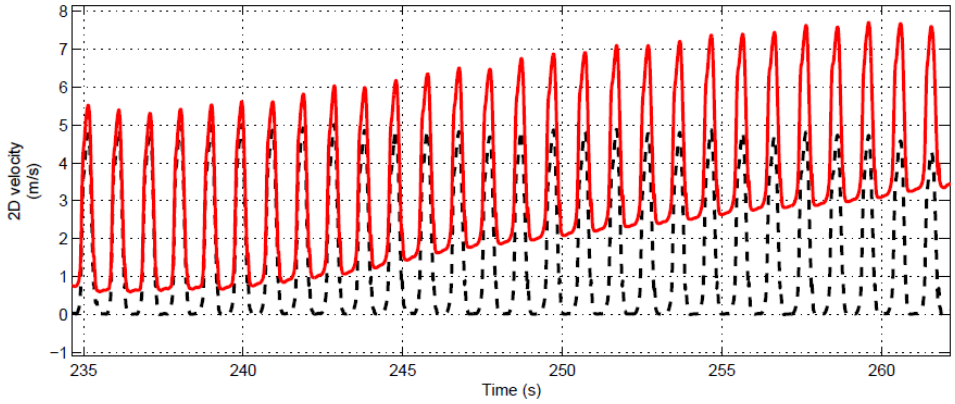

Figure 3.1 2D velocity (along x and y directions) of a shoe-mounted IMU without ZUPTs (solid red line) and with ZUPT correction (dotted black line).

accelerometer and gyro biases [9, 16]. This is illustrated in figure 3.2.

For pedestrian navigation with a shoe-mounted IMU, performing ZUPTs during every stance phase helps to limit the growth of the velocity errors (by resetting them to zero). Applying ZUPTs as measurements into an extended Kalman filter (i.e. the system model is a non-linear function of the states) instead of simply resetting the velocity to zero has additional advantages. The zero-velocity measurements enable the EKF to also estimate the position, attitude and INS instrument errors, which overall increase the position accuracy [1, 15].

ZUPTs do not provide absolute position information but the extended Kalman filter (EKF) builds up the correlations between the velocity and position errors in the corresponding off-diagonal elements of the error covariance matrix, P. This enables the EKF to correct most of the position drift, including 'altitude', since the last ZUPT. For example, if, during the stride phase, the velocity and hence the position are tending to drift in a particular direction, say north, the EKF is able to sense this information during the following ZUPT where it not only corrects the velocity towards zero but also corrects the position to the south [1, 5, 9]. This is illustrated in figure 3.3. During ZUPTs, the EKF corrects the position

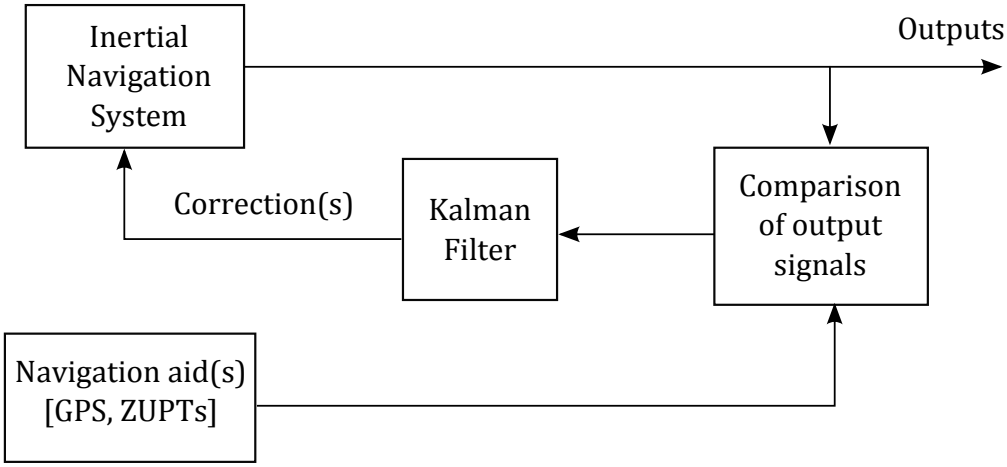

Figure 3.2 Basic principle of an integrated navigation system (adapted from [16]).

drift that has accumulated during the stride phase.

The velocity error dynamics equation in the navigation (n) frame is given by

$$\begin{pmatrix} \dot{\delta v}_x \\ \dot{\delta v}_y \\ \dot{\delta v}_z \end{pmatrix} = \begin{pmatrix} 0 & -a_z^n & a_y^n \\ a_z^n & 0 & -a_x^n \\ -a_y^n & a_x^n & 0 \end{pmatrix} \begin{pmatrix} \delta\phi \\ \delta\theta \\ \delta\psi \end{pmatrix} + \begin{pmatrix} \delta a_x^n \\ \delta a_y^n \\ \delta a_z^n \end{pmatrix} \tag{3.1}$$

where 'dot' represents the time derivative, δv represents the velocity errors in the n-frame, a^n represents the acceleration in the n-frame, $\delta\phi$, $\delta\theta$ and $\delta\psi$ denote the roll, pitch and yaw (heading) errors respectively and δa^n represents the accelerometer biases in the n-frame.

It can be seen from equation 3.1 that, during zero-velocity periods or periods with relatively small accelerations where the magnitude of the acceleration can be neglected with respect to the gravity, the errors in the x and y velocities are related to the pitch and roll errors, respectively, through the acceleration in the z direction which is typically close to gravity (9.8 ms^{-2}). Therefore, the EKF is able to estimate pitch and roll errors and the pitch and roll gyro biases from the zero-velocity measurements. Since the yaw errors are related to the velocity errors through horizontal acceleration, they can be estimated when the person is moving but cannot be estimated from the ZUPTs which are applied during the

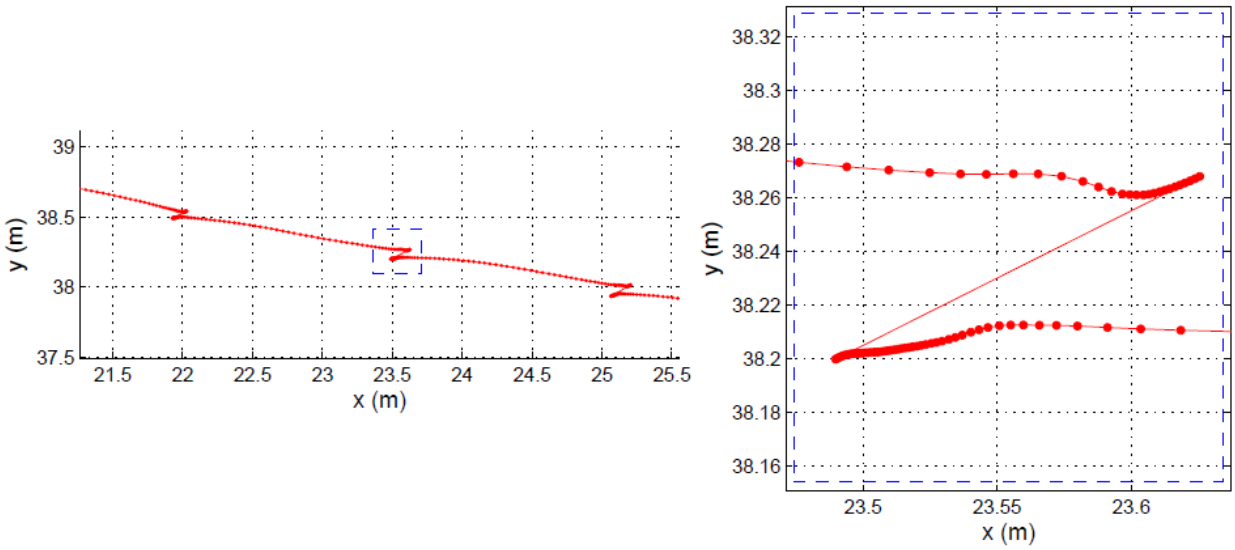

Figure 3.3 Position drift corrected by the EKF during ZUPTs.

stance phase and these are the main error contributing factors in the INS-ZUPTs integrated system [1, 15].

3.2 INS and EKF Formulation using Xsens orientation

A PDR system has been implemented, using the methods described by Foxlin and Jiménez et al [1,6], which makes use of an EKF to combine the INS estimates and ZUPTs applied during every stance phase which is detected using one of the stance phase detection algorithms described in the next section.

The PDR system is initially implemented using the rotation matrix calculated directly in the Xsens MTx's internal software. Therefore, the EKF used in this implementation estimates only the velocity errors, position errors and accelerometer biases.

The main blocks in this implementation, the strapdown INS mechanization process and the Extended Kalman Filter (EKF), are described below. The outputs of the EKF are used in the strapdown INS equations.

3.2.1 Strapdown INS mechanization process

The Xsens MTx IMU measures acceleration and angular rate in the sensor (body) frame(a_k^b and ω_k^b, respectively) at every sample interval Δt at discrete sampling times k. But, for use in the velocity integration step of the navigation equations, the acceleration must be resolved about the same axes as the velocity, that is, the navigation (n) frame. The rotation matrix obtained from the sensor is used to rotate the acceleration in the sensor frame to the n-frame.

The EKF error state vector at time *k+1* after a ZUPT, $\delta x_{k+1} = \begin{pmatrix} \delta r_{k+1}^T & \delta v_{k+1}^T & \delta a_{k+1}^{b^T} \end{pmatrix}^T$, where, δr_{k+1}^T and δv_{k+1}^T denote the estimated position and velocity errors, respectively, and $\delta a_{k+1}^{b^T}$ represents the estimated accelerometer biases, are used to correct the INS outputs. Each of these 3 components has 3 elements each, corresponding to the estimates in the x, y and z axes, respectively.

The INS mechanization process, shown in figure 3.4, consists of the following four steps

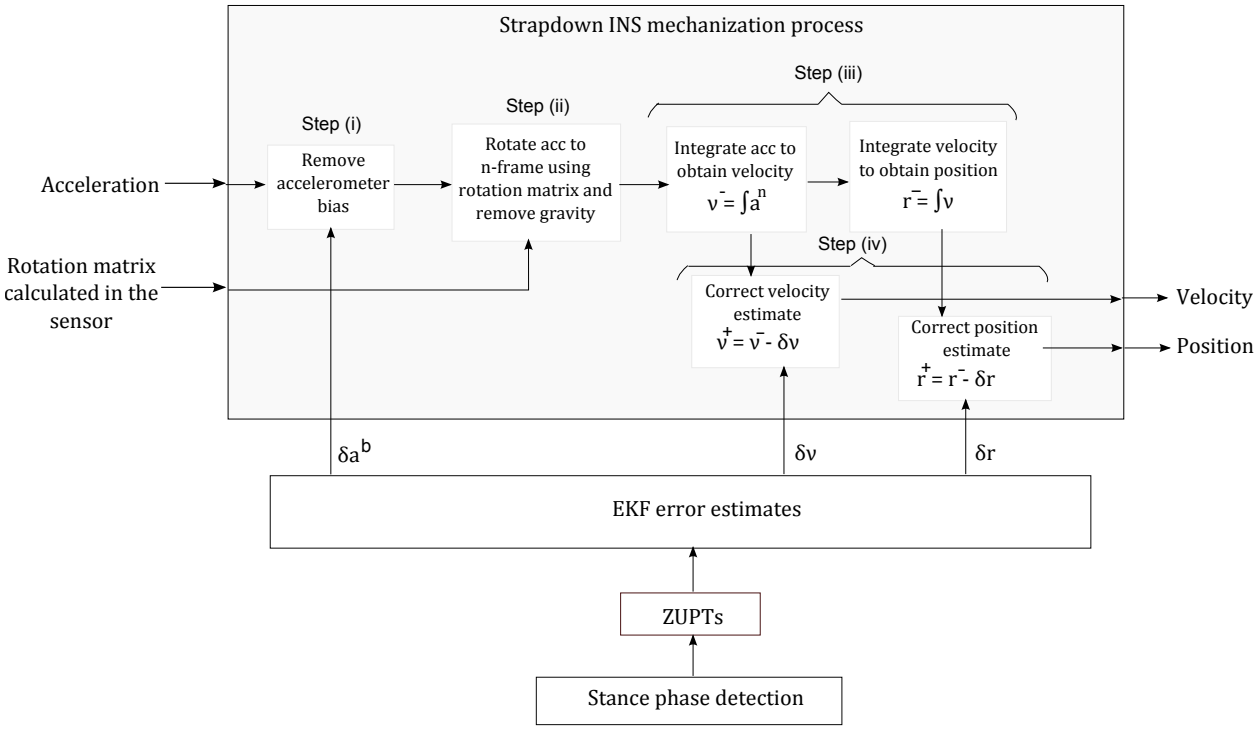

Figure 3.4 Strapdown INS mechanization process

performed during each time interval.

(i) Compensating for accelerometer bias using the EKF accelerometer bias estimates (equations 3.2 and 3.3).

(ii) Rotating the acceleration in the sensor frame to the n-frame using the rotation matrix obtained from the sensor and removing the gravitational component from the z-acceleration in the n-frame (equation 3.4).

(iii) Integrating the acceleration in the n-frame to obtain the velocity estimate (equation 3.5) and integrating the velocity to obtain the position estimate (equation 3.6).

(iv) Correcting the velocity and position based on the EKF error estimates (equations 3.7-3.10).

The four steps given above are explained in detail as follows.

(i) The accelerometer bias estimated by the EKF refers to the bias in the acceleration

measured in the sensor frame, and is hence subtracted from the raw acceleration data obtained from the sensor (equation 3.3). The components 7-9 of the error state vector, δx, contain the accelerometer bias (equation 3.2). The bias compensation is performed using the bias estimated in the previous time interval after a ZUPT.

$$\delta a_k^b = \delta x_k(7:9) \tag{3.2}$$

$$a_{k+1}^{b'} = a_{k+1}^b - \delta a_k^b \tag{3.3}$$

where δa_k^b denotes the accelerometer bias and $a_{k+1}^{b'}$ denotes the bias-compensated acceleration in the sensor frame.

(ii) The acceleration obtained in the previous step is then rotated to the n-frame using the rotation matrix and gravity (9.8 ms^{-2}) is subtracted from the vertical component of the acceleration in the n-frame (equation 3.4).

$$a_{k+1}^n = C_{b_{k+1}}^n . a_{k+1}^{b'} - \begin{pmatrix} 0 & 0 & g \end{pmatrix}^T \tag{3.4}$$

where $C_{b_{k+1}}^n$ denotes the rotation matrix calculated directly in the sensor at time *k+1* which transforms the acceleration in the sensor frame to the n-frame, a_{k+1}^n denotes the acceleration in the n-frame and 'g' denotes gravity, 9.8 ms^{-2}.

(iii) The acceleration in the n-frame obtained in the previous step is integrated to obtain the velocity in the n-frame (equation 3.5). This velocity estimate is then integrated to obtain the position in the n-frame (equation 3.6). 'Trapeze' method is used for the integration.

$$v_{k+1|k} = v_{k|k} + (a_k^n + a_{k+1}^n)\frac{\Delta t}{2} \tag{3.5}$$

$$r_{k+1|k} = r_{k|k} + (v_k + v_{k+1})\frac{\Delta t}{2} \tag{3.6}$$

where $v_{k+1|k}$ and $r_{k+1|k}$ denote the velocity and position in the navigation frame, respectively, prior to the EKF correction at time *k+1*.

(iv) The velocity and position estimates obtained in the previous step are corrected using the respective errors estimated by the EKF after a ZUPT (equations 3.9 and 3.10). The components 1-3 and 4-6 of the error state vector, δx, contain the estimated position and velocity errors, respectively (equations 3.7 and 3.8).

$$\delta r_{k+1} = \delta x_{k+1}(1:3) \tag{3.7}$$

$$\delta v_{k+1} = \delta x_{k+1}(4:6) \tag{3.8}$$

$$v_{k+1|k+1} = v_{k+1|k} - \delta v_{k+1} \tag{3.9}$$

$$r_{k+1|k+1} = r_{k+1|k} - \delta r_{k+1} \tag{3.10}$$

It should be noted that the EKF estimates the errors and the bias only after a ZUPT since the zero-velocity measurements are available only during ZUPTs. After each measurement update, the EKF transfers the error states to the INS and resets the error state vector to zero because those errors are already compensated and incorporated into the INS estimations. Hence, the steps (i) and (iv) are meaningful only after a ZUPT.

3.2.2 The Extended Kalman Filter

Although a Kalman filter can be used to directly estimate the state variables, it is common in inertial navigation systems to instead use a complementary Kalman filter which operates only on the errors in the state variables [17].

As mentioned previously, the EKF estimates the position and velocity errors and the accelerometer bias. The EKF error state vector at time *k+1* is given by

$$\delta x_{k+1} = \begin{pmatrix} \delta r_{k+1}^T & \delta v_{k+1}^T & \delta a_{k+1}^{b^T} \end{pmatrix}^T \tag{3.11}$$

The state-transition matrix, a 9×9 matrix, at time *k+1* given by

$$\Phi_{k+1} = \begin{pmatrix} I_{3\times3} & \Delta t I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & \Delta t C_{b_{k+1}}^n \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{pmatrix} \tag{3.12}$$

where the terms have their usual meanings.

During every time step, the error covariance matrix, P, is propagated using

$$P_{k+1|k} = \Phi_{k+1} P_{k|k} \Phi_{k+1}^T + Q \tag{3.13}$$

where Φ_{k+1} is the above defined state-transition matrix at time *k+1*, $P_{k+1|k}$ is the error covariance matrix at time *k+1* based on measurements received through k and Q is the system noise covariance matrix.

It should be noted that, although the EKF estimates the errors and the bias only after a ZUPT since the zero-velocity measurements are available only during stance phase, the error covariance propagation is performed during each time interval but the error covariance matrix is updated only during a ZUPT [5].

The error state prediction step, $\delta x_{k+1|k} = \Phi_{k+1} \delta x_k$, need not be implemented since the EKF resets the state vector to zero after each measurement update since the errors are already compensated in the INS estimations.

We assume a virtual measurement $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ of the INS velocity during the stance phase. The difference between the INS velocity and the zero-velocity measurement is input to the EKF. Hence, the measurement vector z is given by

$$z = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T - \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T = -\begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T \tag{3.14}$$

The measurement matrix H, a 3×9 matrix, is given by,

$$H = \begin{pmatrix} 0_{3\times3} & -I_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{3.15}$$

The set of equations, 3.14 and 3.15, can be replaced by the set of equations, 3.16 and 3.17.

$$z = \begin{pmatrix} v_x & v_x & v_x \end{pmatrix}^T - \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T \tag{3.16}$$

$$H = \begin{pmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{3.17}$$

The Kalman gain, K, is calculated as follows,

$$K = P_{k+1|k}H^T(HP_{k+1|k}H^T + R)^{-1} \tag{3.18}$$

where R is the measurement noise covariance matrix.

After the EKF receives each measurement, the state vector is updated as follows,

$$\delta x_{k+1} = Kz; \tag{3.19}$$

The error covariance P is then updated using the Joseph form,

$$P_{k+1|k+1} = (I - KH)P_{k+1|k}(I - KH)^T + KRK^T \tag{3.20}$$

The Joseph form of covariance update is used because it ensures greater symmetry about the diagonal in the P matrix [9, 11].

3.3 Stance Phase Detection

In order to apply zero-velocity updates (ZUPTs), it is necessary to identify correctly the periods during which the foot is in contact with the ground (stance phase). Stance phase detection algorithms must be robust enough because position errors, including altitude errors, can significantly increase even if a few ZUPTs are missed. Faulty stance phase detection can also lead to errors in position.

Most of the algorithms implemented so far are based on analyzing the signature of the 3D acceleration or 3D angular rate signal or both. These signals demonstrate a definitive and repetitive pattern for normal walking. Since most of the related works (including this thesis) are based on data collected with a normal walking pattern, even simple algorithms that compare the 3D acceleration and/or 3D angular rate measurements with empirically determined thresholds seem to work well [1, 3, 4, 6, 7, 15]. But, it should be noted that fire-fighters and emergency responders do not have fixed walking patterns. Therefore, more complex approaches will be required to design stance phase detection algorithms that detect zero-velocity periods even with irregular walking patterns which could be results of walking, crawling or running at different speeds, or climbing stairs forward or backward, or jumping, or carrying things or people, or foot slipping, or walking over debris. Stance phase detection for these more complex scenarios is left for future work.

In general, a gait cycle can consist of four gait phases, namely:

(i) Heel-Strike: This phase includes the moment when the foot (heel) just touches the floor. The accelerometer signal demonstrates sudden spikes during this phase (figure 3.5). This phase marks the beginning of the stance phase.

(ii) Stance Phase: This phase occurs when both the heel and toe are on the ground at rest.

(iii) Push-off: During this phase, the heel is off the ground and the toe is on the ground.

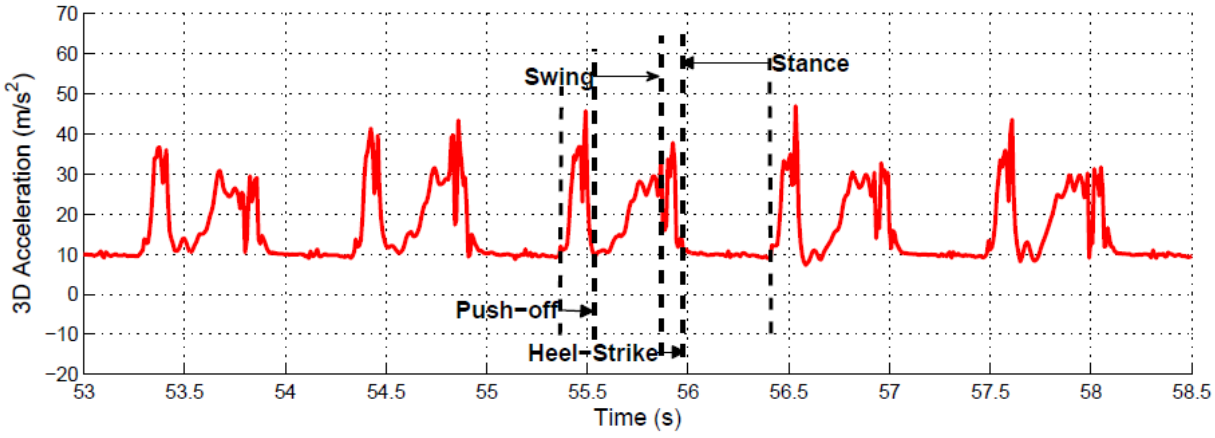

Figure 3.5 Accelerometer signal demonstrating the different gait phases.

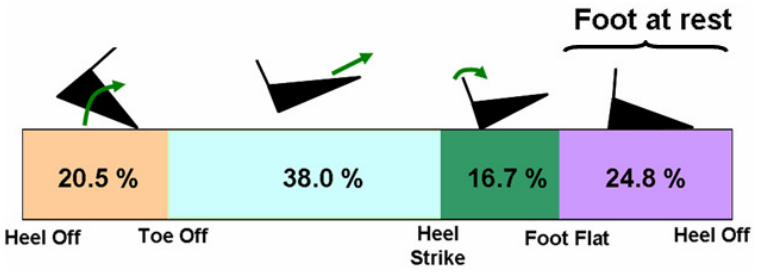

Figure 3.6 Gait Cycle (figure reproduced from [15, Figure 4]).

The accelerometer signal demonstrates sudden spikes during this phase (figure 3.5). Such spikes mark the beginning of the swing phase.

(iv) Swing phase: This phase occurs when both the heel and toe are off the ground. The acceleration pattern during this phase can be used to determine the type of step (forward/backward).

The accelerometer signal demonstrating the different gait phases is shown in figure 3.5. Figure 3.6 shows the percentage of gait cycle that corresponds to each gait phase, push-off (between heel off and toe off), swing (between toe off and heel strike), heel-strike (between heel strike and foot flat) and stance (between foot flat and heel off), under normal walking [15].

Algorithms for stance phase detection Three different algorithms for stance phase detection are implemented in this thesis. They are explained in the following sections.

3.3.1 Using 3D angular rate signal

The first algorithm implemented is based on the stance phase detection method explained by Ojeda and Borenstein [3]. This algorithm uses only the gyro data (ω^b) for stance phase detection. According to this method, it is not necessary to identify correctly the exact start and end times of each stance phase. Instead, it is sufficient to find a single instance, T_s, within each stance phase where all the accelerations and velocities are zero.

The gyro signals are divided into segments (s) of n=50 samples, which correspond to approximately 0.4 seconds of data. The number of samples is chosen such that there is at most one stance phase in each segment. Within each segment, the magnitude of every sample is calculated as follows:

$$\omega_{s,i} = \sqrt{\omega_{x,i}^2 + \omega_{y,i}^2 + \omega_{z,i}^2} \tag{3.21}$$

where 's' denotes the segment and 'i' the sample within the segment 's' ($1 \leq i \leq n$)

Next, for each segment, all the elements of ω_s which are smaller than 0.6 (empirically determined threshold) are copied into a new array, ω_T

$$\omega_{T,i} = \begin{cases} \omega_{s,i} & \text{if } \omega_{s,i} < 0.6 \\ \text{some large number, K} & \text{otherwise} \end{cases} \tag{3.22}$$

If all elements $\omega_{T,i} = K$ for a particular segment, then we conclude that there was no stance phase period in that segment, otherwise if one or more elements $\omega_{T,i} \neq K$, the least-valued element of ω_T is chosen as the instance, T_s, where all the accelerations and velocities are zero within the stance phase in that segment and ZUPT is performed at that instance.

For a segment, if one or more elements $\omega_{T,i} \neq K$, then

$$T_s = \min(\omega_T) \tag{3.23}$$

The above procedure is repeated for every segment.

3.3.2 Using 3D acceleration, 3D angular rate and local acceleration variance

This algorithm is based on the stance phase detection method explained by Godha et al and Jiménez et al [6,15]. In order to make the detection process robust enough, a multi-condition algorithm, based on 3D acceleration (a^b), 3D angular rate (ω^b) and local acceleration variance, is implemented.

The three conditions (*C1, C2, C3*) which have to be satisfied simultaneously in order to detect a stance phase are (the thresholds may be different for other implementations):

(i) The magnitude of the acceleration at time k, $|a_k|$, computed as follows must be between two empirically determined thresholds th_1=9 ms^{-2} and th_2=11 ms^{-2}

$$|a_k| = \sqrt{a_{x_k}^2 + a_{y_k}^2 + a_{z_k}^2} \tag{3.24}$$

Therefore,

$$C1 = \begin{cases} 1 & \text{if } th_1 \leq |a_k| \leq th_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

(ii) The local acceleration variance at time k, which highlights the foot activity and removes gravity, computed as follows must be below a pre-determined threshold th=15 m^2s^{-4}

$$\sigma_{a_k}^2 = \frac{1}{2s+1} \sum_{j=k-s}^{k+s} (a_j - \overline{a_k})^2 \tag{3.26}$$

where, the local mean acceleration value at time k, $\overline{a_k}$ is computed by

$$\overline{a_k} = \frac{1}{2s+1} \sum_{q=k-s}^{k+s} a_q \tag{3.27}$$

where 's' is the size of the averaging window (s=3 samples).

Therefore,

$$C2 = \begin{cases} 1 & \text{if } \sigma^2_{a_k} < th \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

(iii) The magnitude of the angular rate at time k, $|\omega_k|$, computed as follows must be below a pre-determined threshold, th_ω=1.5 $rads^{-1}$

$$\omega_k = \sqrt{\omega^2_{x_k} + \omega^2_{y_k} + \omega^2_{z_k}} \tag{3.29}$$

Therefore,

$$C3 = \begin{cases} 1 & \text{if } |\omega_k| < th_\omega \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

In addition to the 3D acceleration and moving acceleration variance (conditions *C1* and *C2*), another condition is used by Godha et al [15], which maintains a minimum time window between the detected stance phase and the time for the next ZUPT, to prevent any faulty stance phase detection.

In the algorithm described by Jiménez et al [6], all the three conditions mentioned above must be satisfied for stance phase detection (a logical 'AND' is used on the conditions) and the result is filtered out using a median filter with a neighbouring window of 11 samples in total to prevent any faulty stance phase detection. The median filter runs through the samples entry by entry and replaces each entry with the median of the neighbouring samples

(by definition, the median of an odd number of entries is the value of the middle entry when the entries are arranged in ascending or descending order. For an even number of entries, the median is the mean of the two middle entries when the entries are arranged in ascending or descending order).[2] If there is a faulty stance phase detection in one of the entries, say there is a logical '1' in the 11 samples where the rest are logical '0's, then the logical '1' is replaced by a logical '0' by the median filter, thereby correcting the faulty stance phase detection.

In addition to the above, a still phase (corresponding to a non-walking stationary foot) is also detected when only stance samples are detected in a window larger than 2 seconds. Still phase detection can help in the initialization and alignment of the INS. (Initial position and velocity must be provided from an external source before an INS can be used to provide measurements in a navigation system, which is known as the initialization process. Attitude initialization is also known as alignment (initial attitude can either be provided by an external source or calculated using the gyros if the inertial sensors are of good quality) [9]). When a person has not moved for a few seconds (still phase), INS alignment can be performed and the last known position may be stored and used for initialization.

The results of the multi-condition stance phase detection algorithm described by Jiménez et al [6], is shown in figure 3.7. $Cond_1$, $Cond_2$ and $Cond_3$ in the figure refer to the conditions on the 3D acceleration, local acceleration variance and 3D angular rate (*C1*, *C2* and *C3*), respectively, and $Stance_{median}$ refers to the result obtained after using the median filter.

In this thesis, only the conditions *C1*, *C2* and *C3* explained above are used for stance phase detection.

[2]Median. http://en.wikipedia.org/wiki/Median

Median filter. http://en.wikipedia.org/wiki/Median_filter

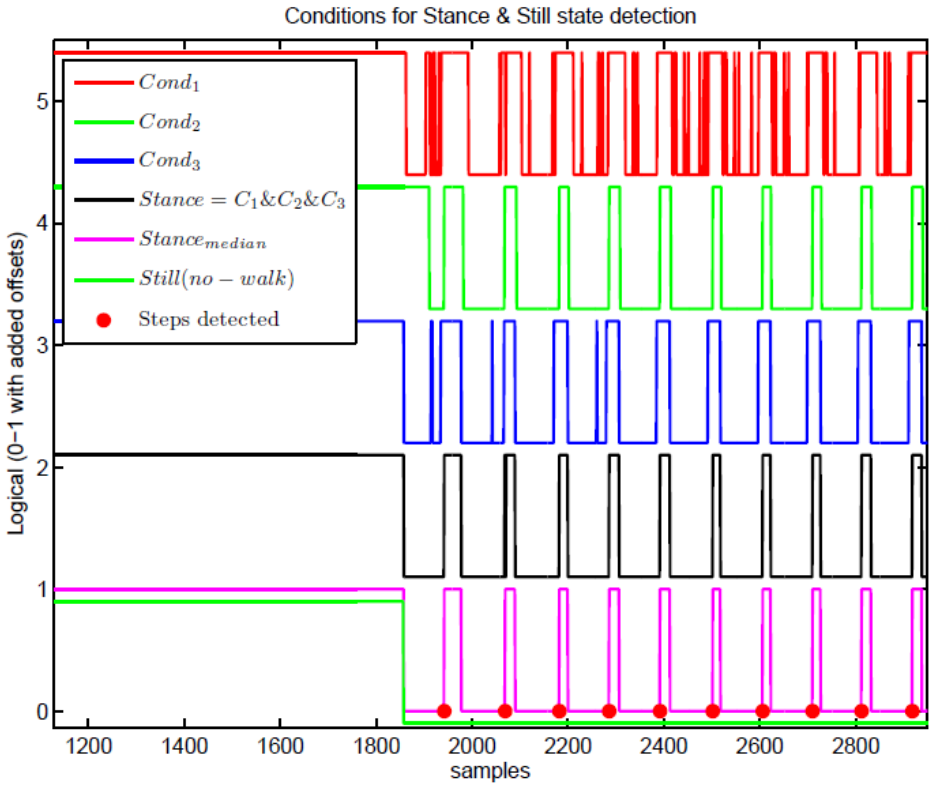

Figure 3.7 Results of the multi-condition stance phase detection algorithm (figure reproduced from [6, Fig. 4]).

3.3.3 Gait phase detection using Hidden Markov Model

This algorithm is based on the gait phase detection method described by Soo Suh and Park [18]. The gait states are modeled as a Markov process and the gait state is estimated using the Hidden Markov Model (HMM) filter.

Formal descriptions of a finite state hidden Markov model and a hidden Markov model filter are given below and then the algorithm is explained [19].

Finite State Hidden Markov Model (Discrete Time)

There is an underlying state process $X_0, X_1, \cdots, X_k$, where k represents the discrete sampling times, and X_k can assume one of the finite set of values, say, $1, 2, \cdots, N$.

The conditional probability of $X_{k+1} = i$, given $X_k = j$, given by $Pr[X_{k+1} = i | X_k = j] = a_{ij}$ is a transition probability and X_k is a Markov process (its future state depends only on the present state and is independent of its past states.[3])

We denote by A, the matrix of state transition probabilities $[a_{ij}]$. The set of transition probabilities should satisfy the following conditions (equations 3.31 and 3.32):

$$a_{ij} \geq 0, \quad 1 \leq i, j \leq N \tag{3.31}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \quad 1 \leq i \leq N \tag{3.32}$$

There is also an output process $Y_0, Y_1, \cdots$ that can consist of infinite values if the observations are continuous, but for convenience lets assume Y_k takes one of a finite set of values, say, $1, 2, \cdots, M$.

It should be noted that, it is only the outputs Y_k and not the states X_k that are visible to an external observer, therefore the states are "hidden" from the observer, and hence the name hidden Markov model.

[3]Markov process. http://en.wikipedia.org/wiki/Markov_process

The link between the state process and the output process is given by $Pr[Y_k = m|X_k = n] = c_{mn}$, and we denote by C, the matrix $[c_{mn}]$. The entries of C should satisfy the following conditions (equations 3.33 and 3.34):

$$c_{mn} \geq 0, \quad 1 \leq n \leq N, \quad 1 \leq m \leq M \tag{3.33}$$

$$\sum_{n=1}^{N} c_{mn} = 1, \quad 1 \leq m \leq M \tag{3.34}$$

The two matrices, A and C, with probabilities as their entries describe the hidden Markov model process.

Hidden Markov Model Filter

A hidden Markov model filter calculates the N-vector whose i^{th} entry is the probability of the state X_k being i, given the outputs up to time k, that is $Pr[X_k = i|Y_0, Y_1, \cdots, Y_k]$.

The calculation of the N-vector consists of the following steps:

(i) The probability vector $\Pi_{k+1|k}$, whose i^{th} element is $Pr[X_{k+1} = i|Y_0, Y_1, \cdots, Y_k]$, is calculated using the probability vector $\Pi_{k|k}$, whose i^{th} element is $Pr[X_k = i|Y_0, Y_1, \cdots, Y_k]$ as follows (equation 3.35):

$$\Pi_{k+1|k} = A\Pi_{k|k} \tag{3.35}$$

(ii) The probability vector $\Pi_{k+1|k+1}$ is updated using the probability vector calculated above, $\Pi_{k+1|k}$, and the output at time $k+1$, Y_{k+1} (equation 3.36).

$$\Pi_{k+1|k+1} = \frac{1}{[1 \cdots 1]C_{y_{k+1}}\Pi_{k+1|k}} C_{y_{k+1}}\Pi_{k+1|k} \tag{3.36}$$

where $C_{y_{k+1}} = diag(c_{p1}, \cdots, c_{pN})$ when $Y_{k+1} = p$.

Algorithm for gait phase detection

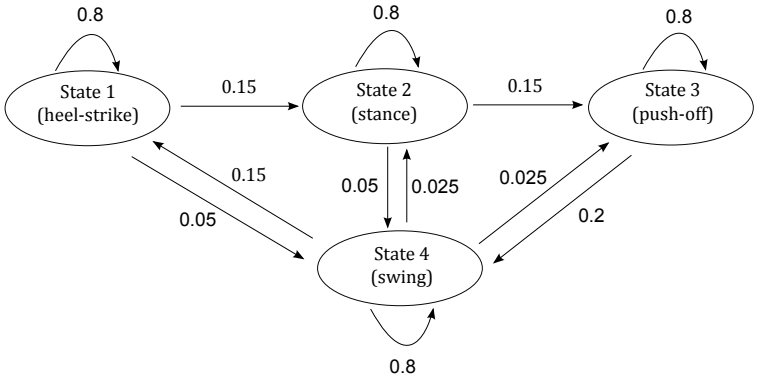

Figure 3.8 State transition diagram

The gait state estimation $(X_0, X_1, \cdots, X_k)$ is modeled as a Markov process and the gait state X_k can assume one of the four gait states 1, 2, 3 and 4 corresponding to heel-strike, stance, push-off and swing phases, respectively.

The state transition probability matrix A is assumed as follows (equation 3.37) and the state transition diagram is given in figure 3.8:

$$A = \begin{pmatrix} 0.8 & 0 & 0 & 0.15 \\ 0.15 & 0.8 & 0 & 0.025 \\ 0 & 0.15 & 0.8 & 0.025 \\ 0.05 & 0.05 & 0.2 & 0.8 \end{pmatrix} \tag{3.37}$$

where each entry a_{ij} represents $Pr[X_{k+1} = i | X_k = j]$.

The 3D acceleration and angular rate measurements are used to define the output process, Y_k. The 3D acceleration and 3D angular rate outputs are quantized into 2 levels each, as shown in table 3.1.

Therefore, there are totally $2 \times 2 = 4$ combinations and each combination is assigned an output value from 1 to 4 as shown in table 3.2.

The link between the state process X_k and output process Y_k, defined by matrix C is assumed as follows (equation 3.38):

Sensors	1	2
3D acceleration	$9 \leq \lvert a_k^b \rvert \leq 11$	$\lvert a_k^b \rvert < 9$ or $\lvert a_k^b \rvert > 11$
3D angular rate	$\lvert \omega_k^b \rvert < 1.5$	$\lvert \omega_k^b \rvert \geq 1.5$

Table 3.1 Sensor outputs quantization

Y_k	3D acceleration	3D angular rate
1	1	1
2	1	2
3	2	1
4	2	2

Table 3.2 Output (Y_k) values

$$C = \begin{pmatrix} 0.01 & 0.9 & 0.01 & 0 \\ 0.49 & 0.1 & 0.49 & 0 \\ 0.01 & 0 & 0.01 & 0 \\ 0.49 & 0 & 0.49 & 1 \end{pmatrix} \tag{3.38}$$

where each entry c_{mn} represents $Pr[Y_k = m | X_k = n]$. For example, the first row of C, corresponding to output $Y_k = 1$, $\begin{pmatrix} 0.01 & 0.9 & 0.01 & 0 \end{pmatrix}$, states that given $Y_k = 1$, the probability that the gait state is 2 is the highest (0.9).

The HMM filter is used to estimate the gait states at discrete sampling times k using the equations 3.35 and 3.36. The HMM filter uses the outputs up to time k, $Y_0, Y_1, \cdots, Y_k$, to provide the best possible statement regarding the value of the gait state at time k. For example, if probability vector calculated at time k, $\Pi_{k|k} = \begin{pmatrix} 0.1 & 0 & 0.67 & 0.23 \end{pmatrix}^T$, given the outputs $Y_0, Y_1, \cdots, Y_k$, then the gait state at time k takes the value 3, because it has the highest probability (0.67).

Zero-velocity updates are performed every time the gait state takes the value 2 (stance phase).

The algorithm implemented above is a simple one using only the 3D acceleration and angular rate sensor measurements for defining the output process Y_k. But this algorithm could be used in more complex scenarios. Generally, for gait analysis (detecting and classifying different walking behaviors such as forward/backward walking, stair climbing and so on), a number of inertial sensors are attached to different parts of the body. The different output patterns obtained from these sensors are used for the gait analysis [20]. The outputs from these sensors could be used to define the output process Y_k in the above algorithm for gait phase detection. Also, the probabilities in the A and C matrices could be determined empirically.

3.4 Results of the three stance phase detection algorithms

The results of the three different algorithms for stance phase detection described in section 3.3 are presented in this section. The PDR system was implemented in accordance with the implementation described in section 3.2 and same values of the extended Kalman filter (EKF) parameters were used for all the implementations presented in this section.

The algorithms explained in sections 3.3.1, 3.3.2 and 3.3.3 will be referred to as algorithms 1, 2 and 3, respectively, in this section.

Sample plots of 3D acceleration, angular rate and velocity signals highlighting the zero-velocity periods detected by each of the algorithms 1, 2 and 3 are presented in section 3.4.1. The plots of the 2D traces and altitudes obtained for the different data sets using the algorithms 1, 2 and 3, the total distances travelled and the altitude variations in each case are presented in section 3.4.2. Observations are reported within each of the respective subsections.

3.4.1 Plots with highlighted zero-velocity periods

Sample plots of 3D acceleration, angular rate and velocity signals highlighting the zero-velocity periods detected by the algorithms 1, 2 and 3 are shown in the figures 3.9, 3.10 and 3.11, respectively.

Although very little can be said about the performance of the stance phase detection algorithms by looking at the plots 3.9, 3.10 and 3.11, a general idea can be obtained regarding the basic working of the algorithms and it can be seen if the algorithms are doing what they were intended or designed to do.

In figure 3.9, the detected zero-velocity periods are represented by lines whereas in fig-

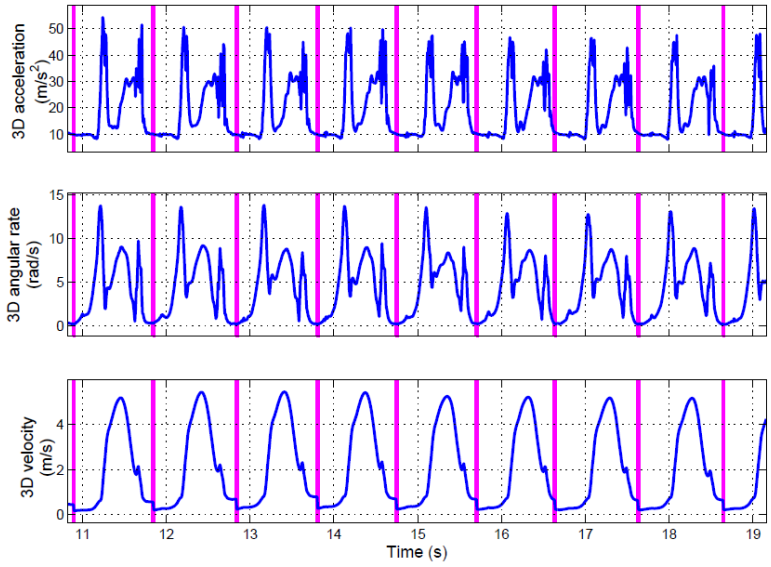

Figure 3.9 3D acceleration, angular rate and velocity signals highlighting the zero-velocity periods detected by algorithm 1

ures 3.10 and 3.11, the detected zero-velocity periods are shaded regions. This is because algorithm 1 is designed to detect only a single instance within each stance phase where the angular rate has the lowest value. Since ZUPTs are performed only during those instances, there may be a slight drift in the velocity during the rest of the stance phase which is left uncorrected in algorithm 1. This is illustrated in figure 3.12. Algorithms 2 and 3 are designed to detect entire zero-velocity periods.

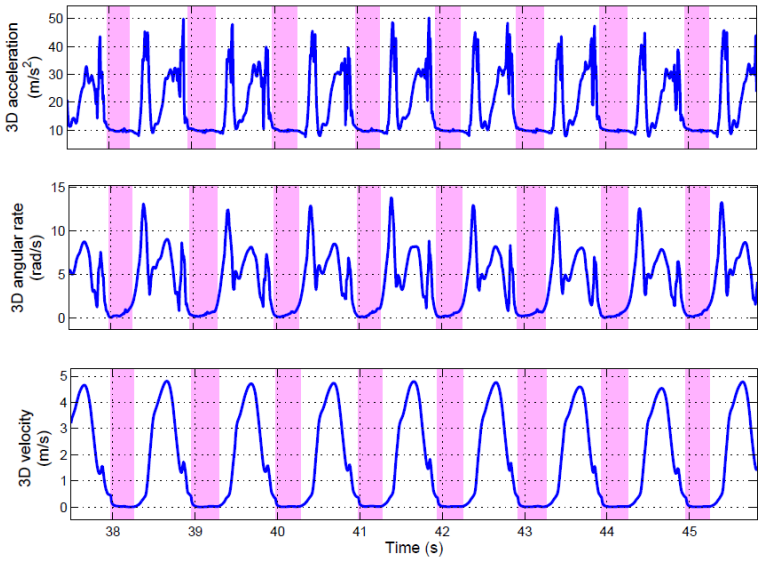

Figure 3.10 3D acceleration, angular rate and velocity signals highlighting the zero-velocity periods detected by algorithm 2

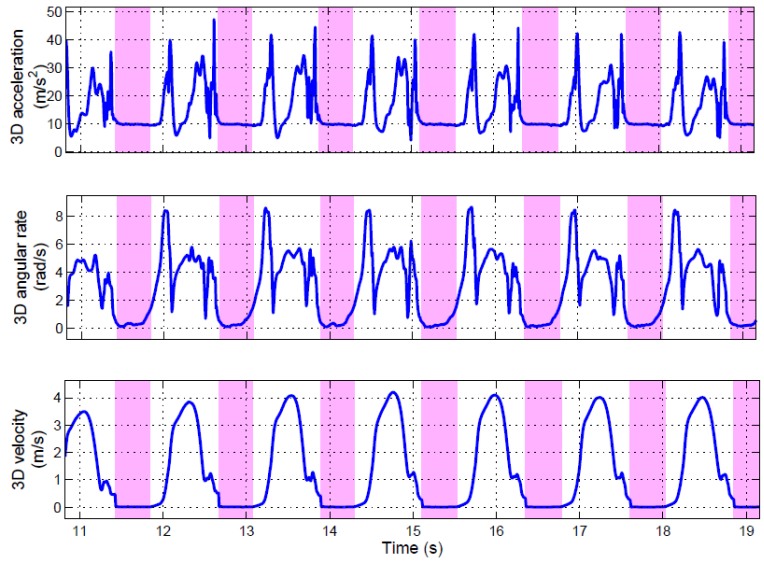

Figure 3.11 3D acceleration, angular rate and velocity signals highlighting the zero-velocity periods detected by algorithm 3

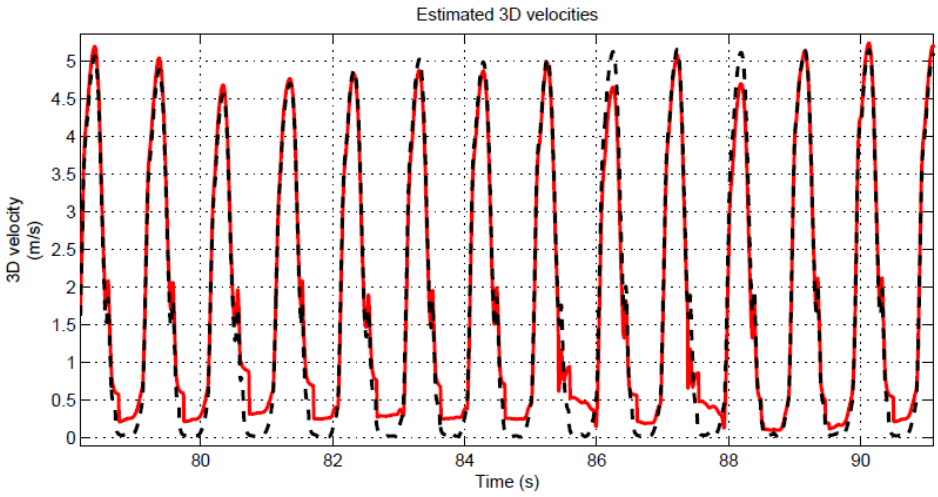

Figure 3.12 3D velocity plots obtained using algorithm 1 (solid red line) and algorithm 2 (dotted black line)

3.4.2 Plots of 2D traces and altitude

Plots of the 2D traces and altitudes obtained using the algorithms 1, 2, and 3 are presented in this section. The plots are ordered by the data sets used in this thesis (section 2.4). The plots of the 2D traces and altitudes obtained using algorithms 1, 2 and 3 are illustrated using solid black, solid green and dotted red lines, respectively.

A. A variety of data has been considered for analysis including different trail topologies ('Long corridor', 'T1 and T2', 'T3 and T4' and Biba workshop), different users ('L1', 'L2' and 'L3' in 'Long corridor'), different durations (ranging from 2.5 to 9 minutes), different walking speeds (T3 (slow) and T4 (fast)), different sensor output settings (not using magnetometer data and giving more weight to accelerometer data relative to magnetometer data in orientation computation (T1 and T2, respectively), different sampling frequencies (100Hz for Twente and Biba data sets to 120Hz for Long Corridor data sets), data with magnetic interference (Biba workshop) and data recorded with stairs (Biba workshop).

Long corridor: The plots of the 2D traces and altitudes obtained for the 'Long corridor' data sets (section 2.4.1) are shown in figures 3.13-3.16 where 'L1', 'L2' and 'L3' represent the data recorded for three different users and 'L4' represents the data set with a longer duration.

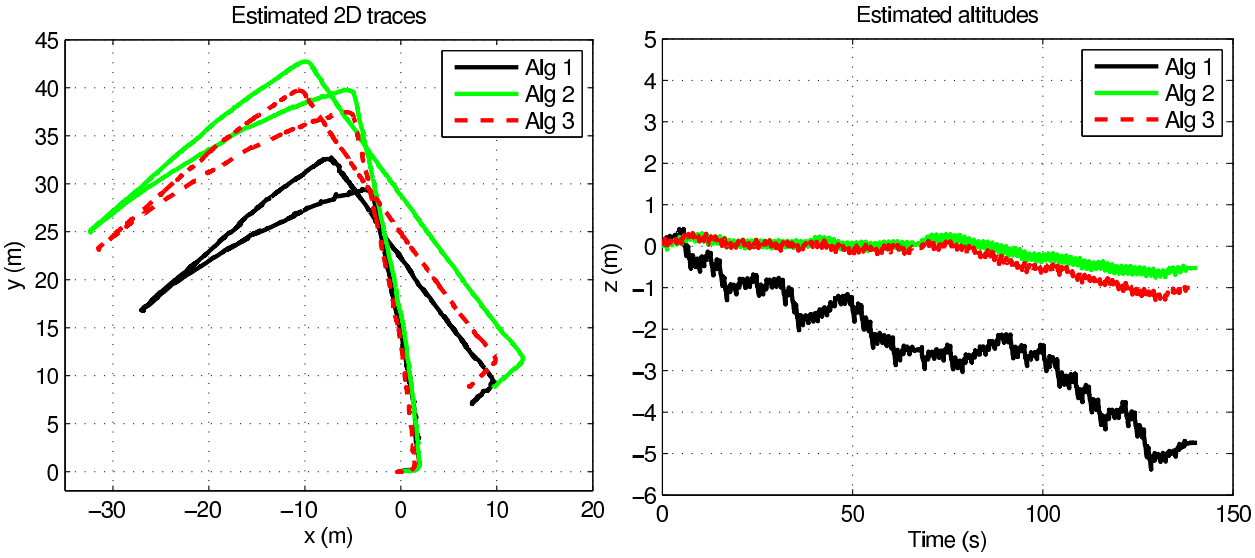

Figure 3.13 Plots of 2D traces and altitude for L1

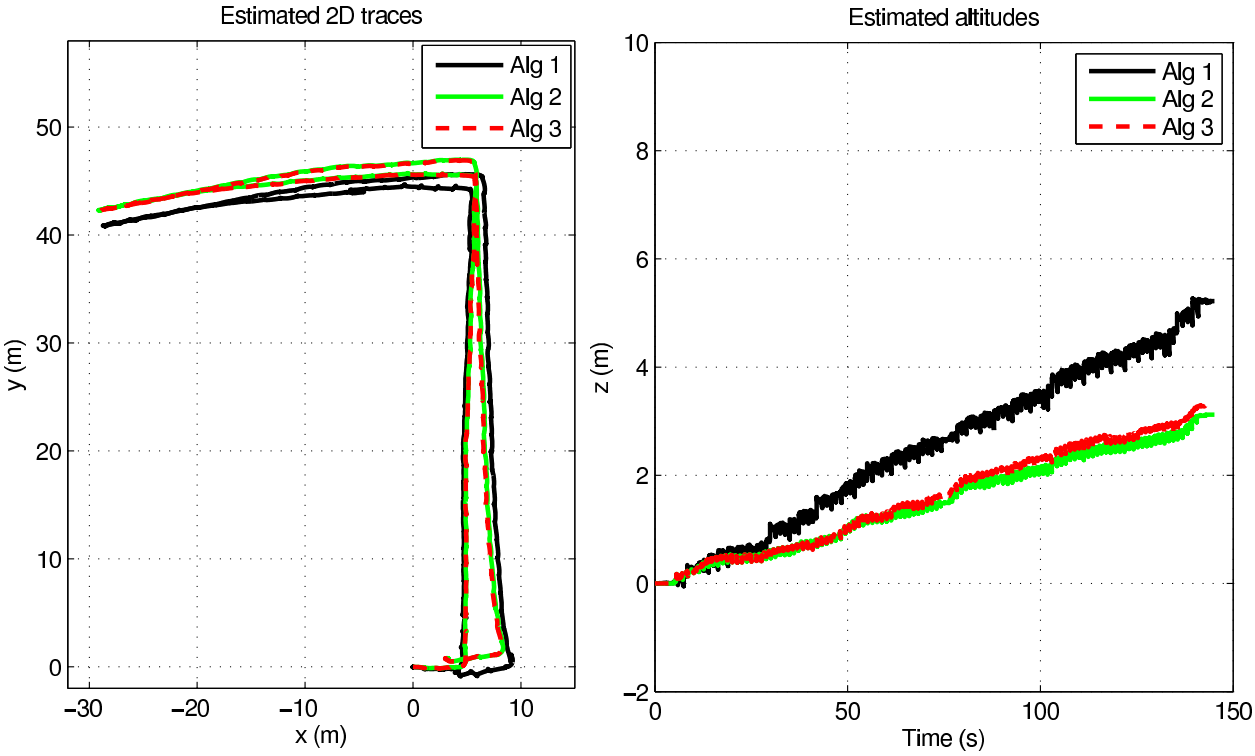

Figure 3.14 Plots of 2D traces and altitude for L2

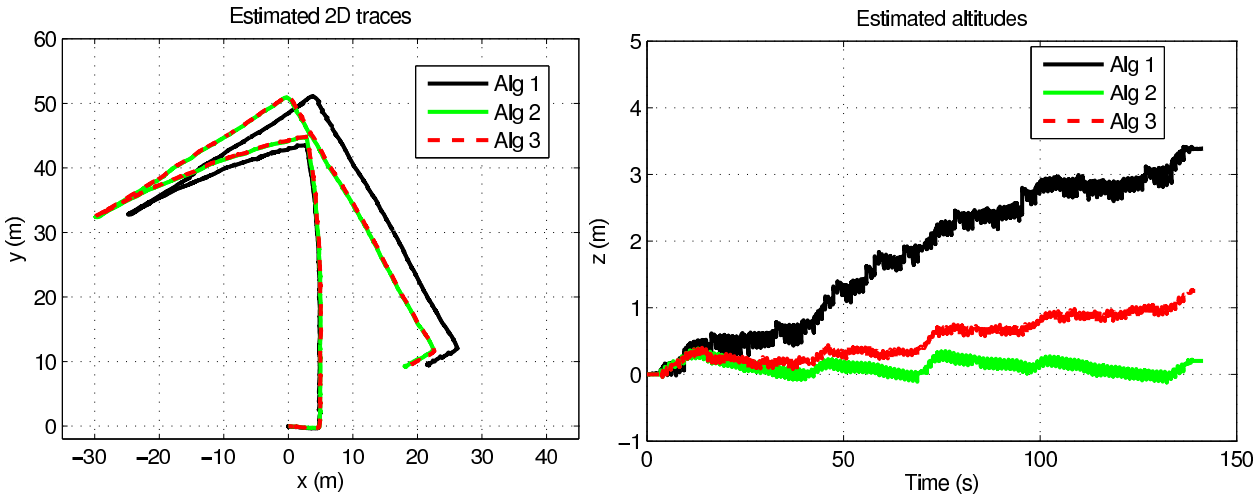

Figure 3.15 Plots of 2D traces and altitude for L3

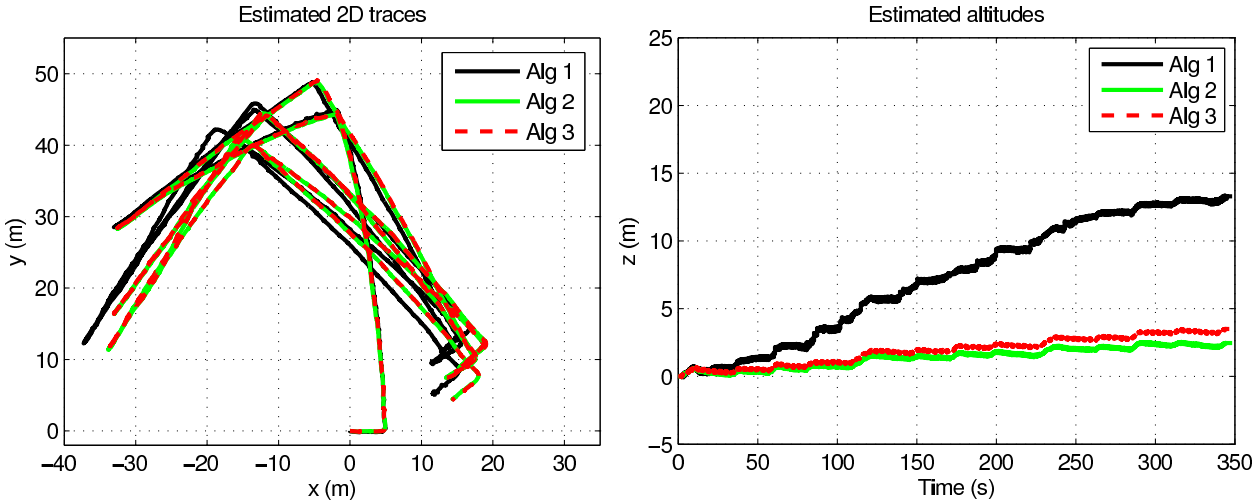

Figure 3.16 Plots of 2D traces and altitude for L4

Data recorded in University of Twente: The plots of the 2D traces and altitudes obtained for the data sets recorded in University of Twente (section 2.4.2) are shown in figures 3.17-3.20.

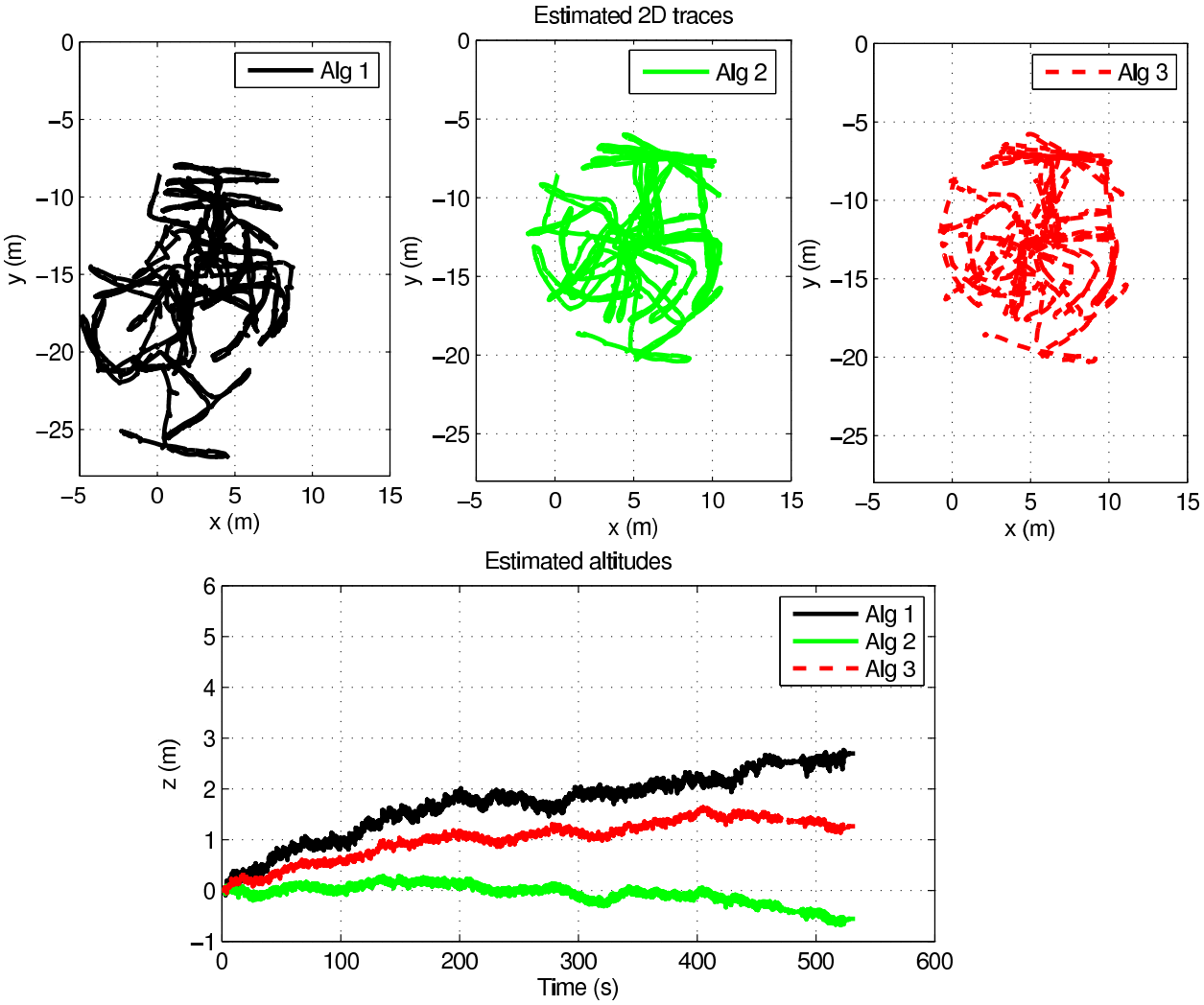

Figure 3.17 Plots of 2D traces and altitude for T1

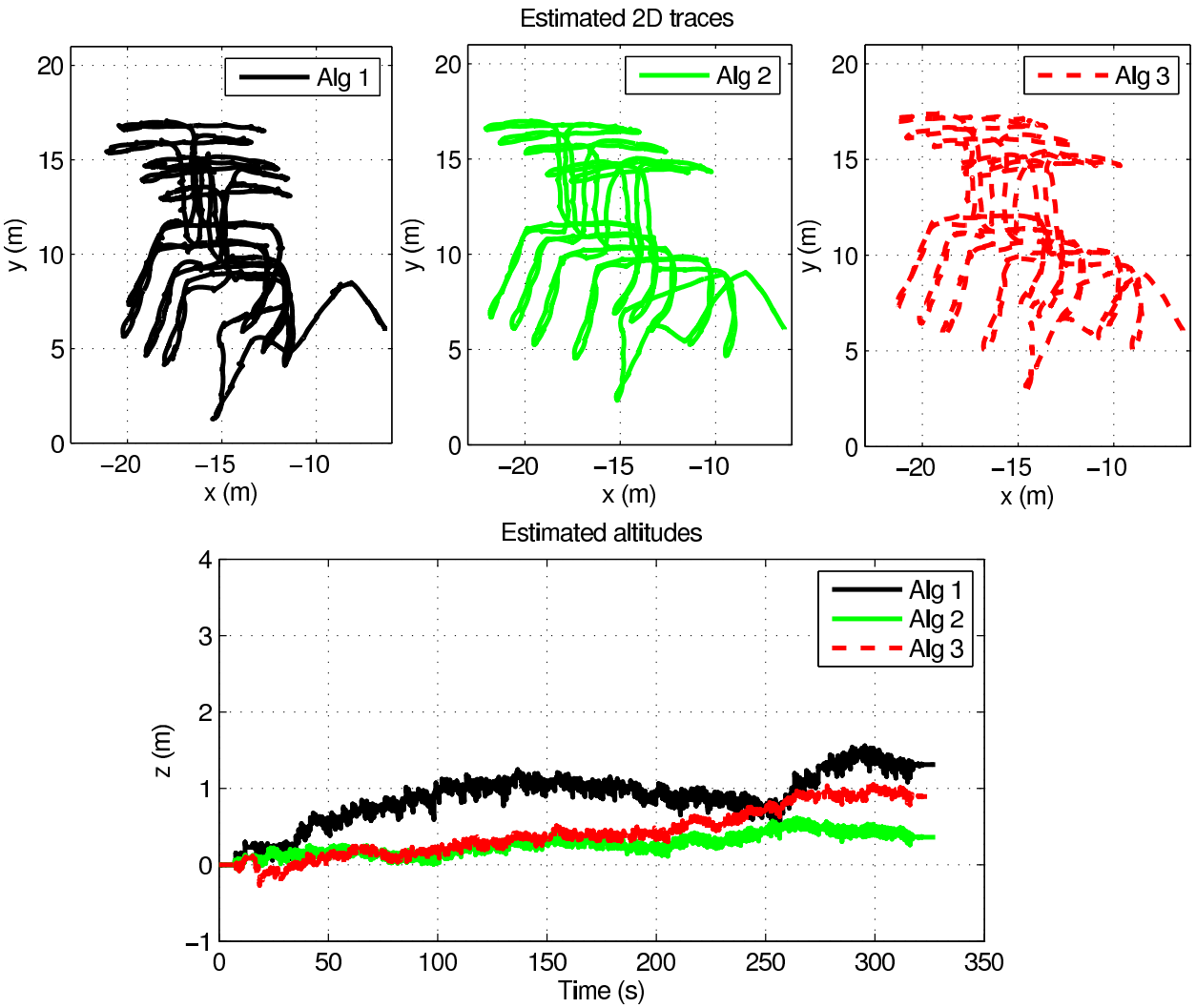

Figure 3.18 Plots of 2D traces and altitude for T2

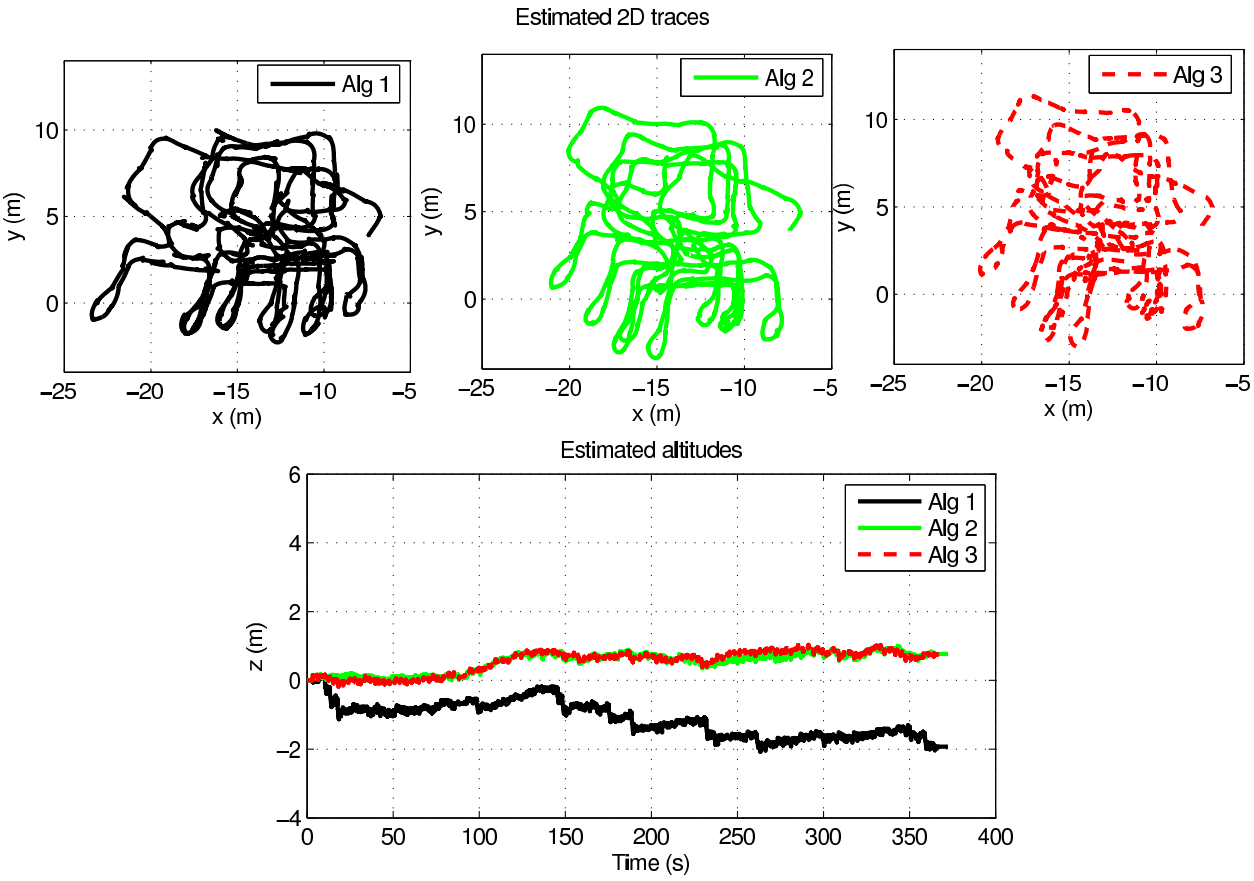

Figure 3.19 Plots of 2D traces and altitude for T3

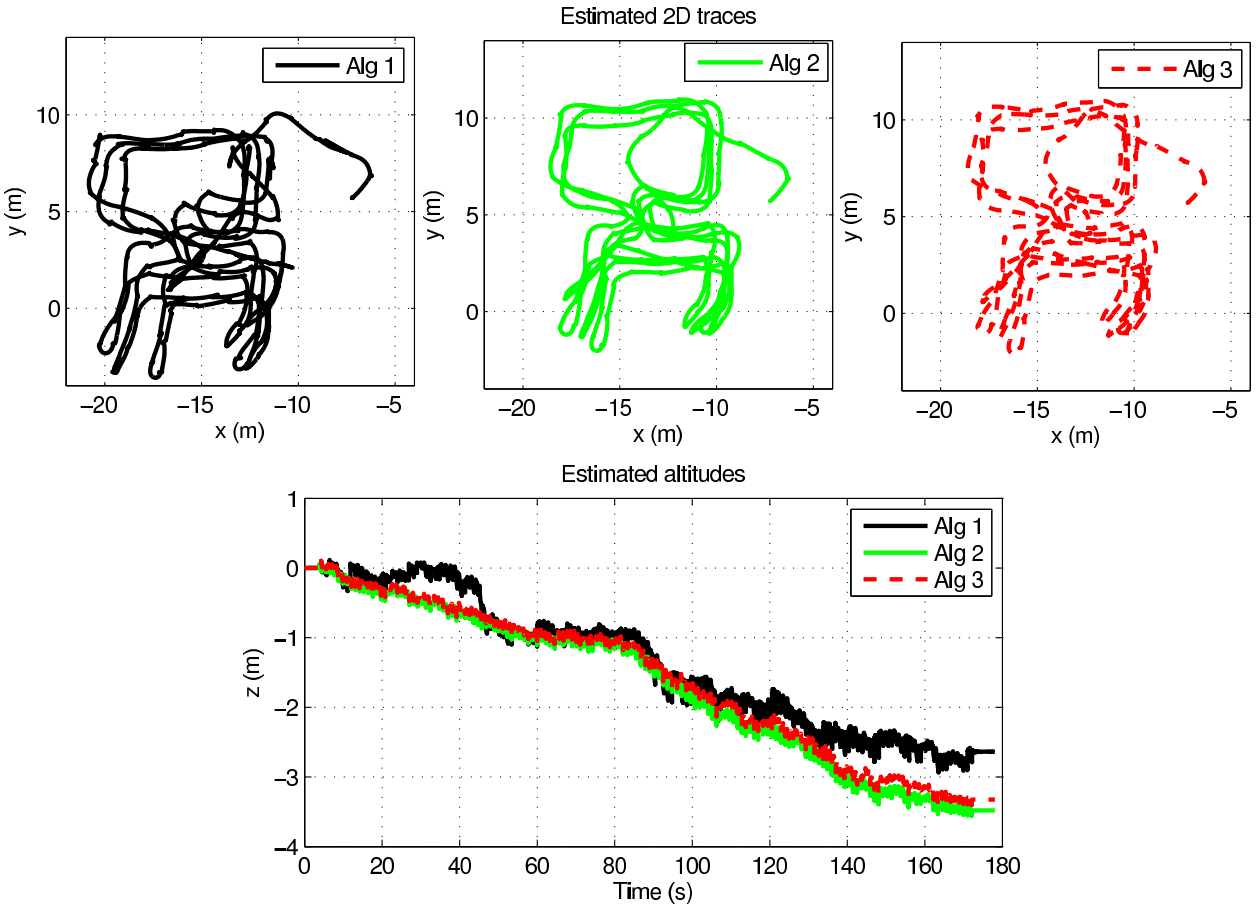

Figure 3.20 Plots of 2D traces and altitude for T4

Biba workshop: The plots of the 2D traces and altitudes obtained for the data sets recorded in Biba workshop (section 2.4.3) are shown in figure 3.21.

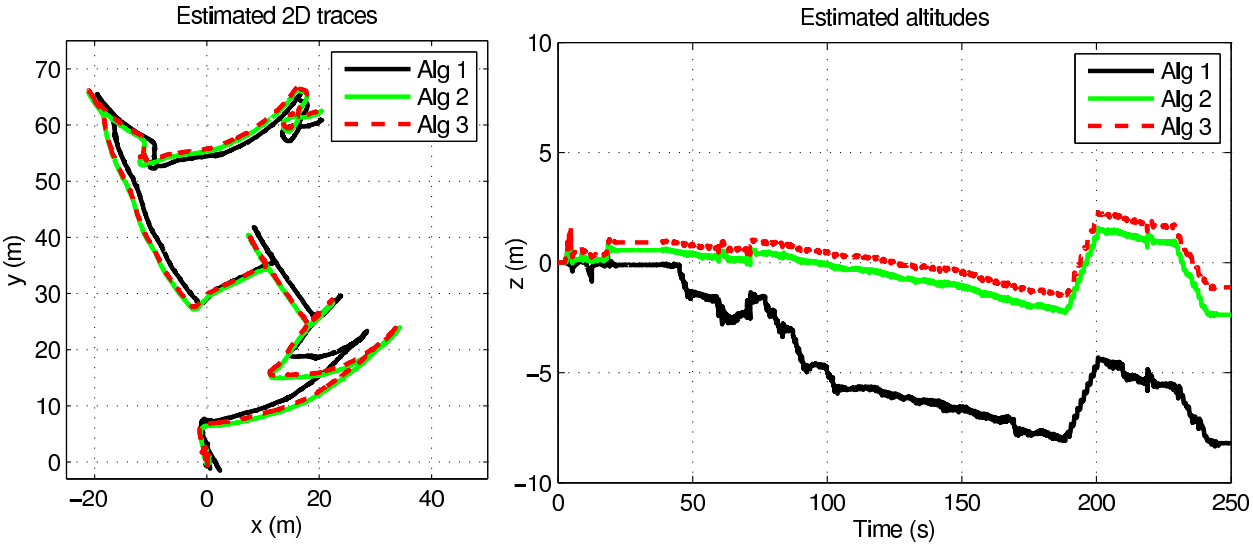

Figure 3.21 Plots of 2D traces and altitude for Biba

B. The groundtruth distances and distances travelled (in metres) using the algorithms 1, 2 and 3 for the different data sets is given in table 3.3. The groundtruth distances for the data recorded in University of Twente were obtained using the Ubisense position estimates and they may not reflect the actual groundtruth distances.

The altitude variations for the different data sets (without stairs) using algorithms 1, 2 and 3 is given in table 3.4.

3.4.3 Discussion

Some general observations, independent of the algorithms used, were recorded. It can be observed from the T1 and T2 plots (figures 3.17 and 3.18) that for those particular Twente data sets, the orientation calculated when accelerometer data is assigned more weight relative to magnetometer data is better than the orientation obtained without using magnetometer data.

	Groundtruth	Algorithm 1		Algorithm 2		Algorithm 3	
Data set	Distances (m)	Distances (m)	Error(%)	Distances (m)	Error(%)	Distances (m)	Error(%)
L1	169.9	232.73	36.98	176.52	3.9	179.44	5.62
L2	169.9	202.56	19.22	178.82	5.25	179.05	5.39
L3	169.9	212.47	25.06	179.33	5.55	182.15	7.12
L4	508.8	615.27	20.93	533.55	4.86	536.90	5.52
T1	665.67*	595.15	n/a	495.35	n/a	494.35	n/a
T2	399.93*	359	n/a	298.13	n/a	298.12	n/a
T3	379.85*	401.06	n/a	326.03	n/a	325.43	n/a
T4	211.06*	270.42	n/a	237.79	n/a	236.86	n/a
Biba	not known	370.73	n/a	301.90	n/a	300.29	n/a

Table 3.3 Groundtruth distances and distances travelled using the algorithms 1, 2 and 3 and their respective percentage errors. Starred groundtruth distances were collected using Ubisense.
(Ubisense distance estimates do not reflect the actual groundtruth distances. Therefore, the calculated percentage errors using Ubisense estimates as groundtruth will not represent the true percentage errors.)

It can be observed from the T3 and T4 plots (figures 3.19 and 3.20) that for those particular Twente data sets, the sensor measurements obtained for T4 (fast walking-approximately 1.34 ms^{-1}) are better than the measurements obtained for T3 (slow walking-approximately 0.87 ms^{-1}). Almost all the traces capture details and most of the individual segments of the recorded paths are very accurate (for example, the spiral staircases in the Biba workshop plot 3.21).

The 2D traces obtained using the different stance phase detection algorithms look identical for almost all the data sets. From table 3.3, it can be seen that the total distance travelled is highest for algorithm 1. As mentioned earlier, since ZUPTs are applied only once during a stance phase in algorithm 1, there is a small velocity drift during the rest of the stance phase which contributes to the position drift.

Data set	Duration (min)	Algorithm 1	Algorithm 2	Algorithm 3
		Altitude value at end of trace $(m)^*$	Altitude value at end of trace $(m)^*$	Altitude value at end of trace $(m)^*$
L1	2.34	4.74	0.53	1.00
L2	2.42	5.22	3.12	3.30
L3	2.36	3.39	0.20	1.25
L4	5.8	13.30	2.46	3.51
T1	8.88	2.70	0.55	1.27
T2	5.46	1.31	0.36	0.89
T3	6.21	1.93	0.77	0.75
T4	2.96	2.63	3.48	3.32

Table 3.4 The altitude variations for the different data sets (without stairs) using algorithms 1, 2 and 3. The starred distances in the table are absolute values. (The altitude traces obtained for almost all the implementations are either continuously decreasing or continuously increasing. Therefore, the (absolute) altitude value at the end of the trace would represent the maximum variation from the true altitude ('0') for each implementation.)

Although algorithms 2 and 3 are not similar, it can be observed from the plots and tables that they perform in a similar way. This is because 3D acceleration and angular rate measurements and the same conditions on them are used in both the algorithms. Algorithm 2 is simple and less computationally intensive compared to algorithm 3 and it can be used for regular walking patterns.

The total distances travelled and the altitude variations for almost all the data sets are greatest for algorithm 1. Algorithms 2 and 3 seem to perform well consistently. For the rest of the implementations in this thesis, algorithm 2 is used for stance phase detection.

3.5 Results of the EKF/ZUPTs formulation using Xsens orientation

The results of the PDR system implementation described in section 3.2 are compared with the results obtained using naive ZUPTs (without EKF) where the velocity is simply reset to zero and results of the comparison are presented in this section. Both the naive ZUPTs and EKF implementations use the rotation matrix calculated in the Xsens MTx IMU.

The stance phase detection algorithm described in section 3.3.2 (algorithm 2) is used for all the implementations in this section. Same values of the extended Kalman filter (EKF) parameters were used for all the results obtained using the PDR system implementation with EKF.

Plots of the 2D traces and altitudes obtained using the implementations with and without EKF are displayed in figures 3.22-3.30. The 2D traces and altitude estimates obtained using the implementations with and without EKF are illustrated using solid green and black lines, respectively.

A table containing the groundtruth distances and distances travelled (in metres) using the implementations with and without EKF for the different data sets is given in table 3.5.

The altitude variations for the different data sets (without stairs) using naive ZUPTs and EKF is given in table 3.6.

The general observations listed in the beginning of the observations part in section 3.4.2 hold true for the results in this section as well. Both the implementations, naive ZUPTs and EKF, use the sensor's orientation. Therefore, the benefits of using an EKF are very few as mentioned earlier (section 3.2). Plots of the 2D traces obtained for the two implementations look almost identical but one significant improvement in the EKF implementation is that the altitude variation is almost negligible in the EKF implementation compared to that in the

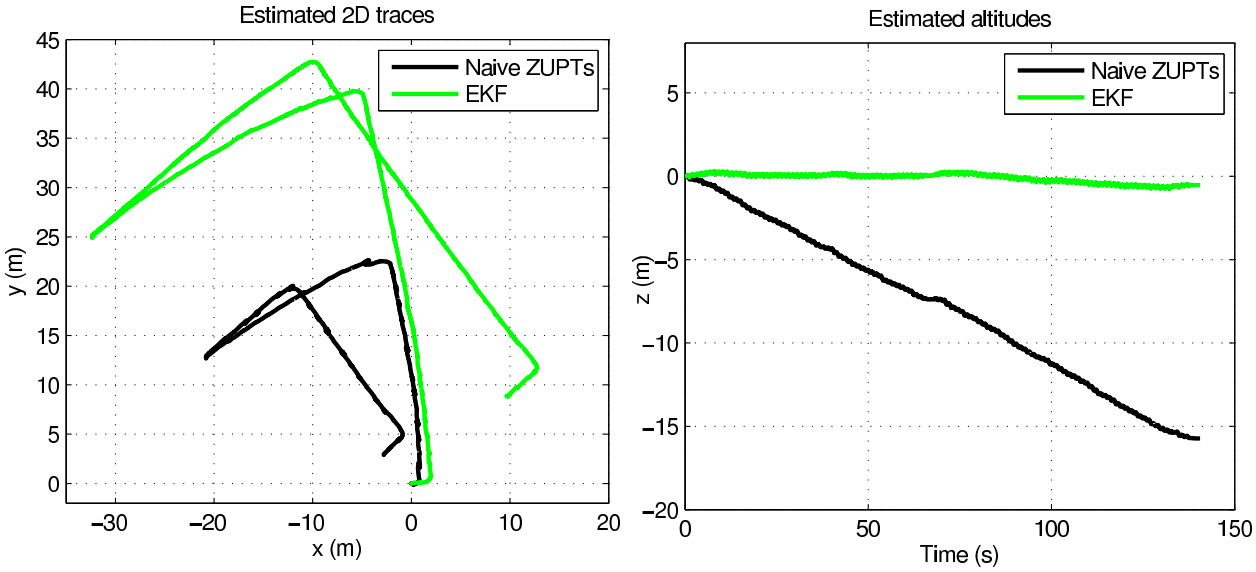

Figure 3.22 Plots of 2D traces and altitude, L1

naive ZUPTs implementation. Even for longer duration plots (figures 3.26 and 3.25), where the maximum altitude variation in the naive ZUPTs implementation is about 22-35 m, the corresponding variation in the EKF implementation is only about 0.5-2.5 m (table 3.6).

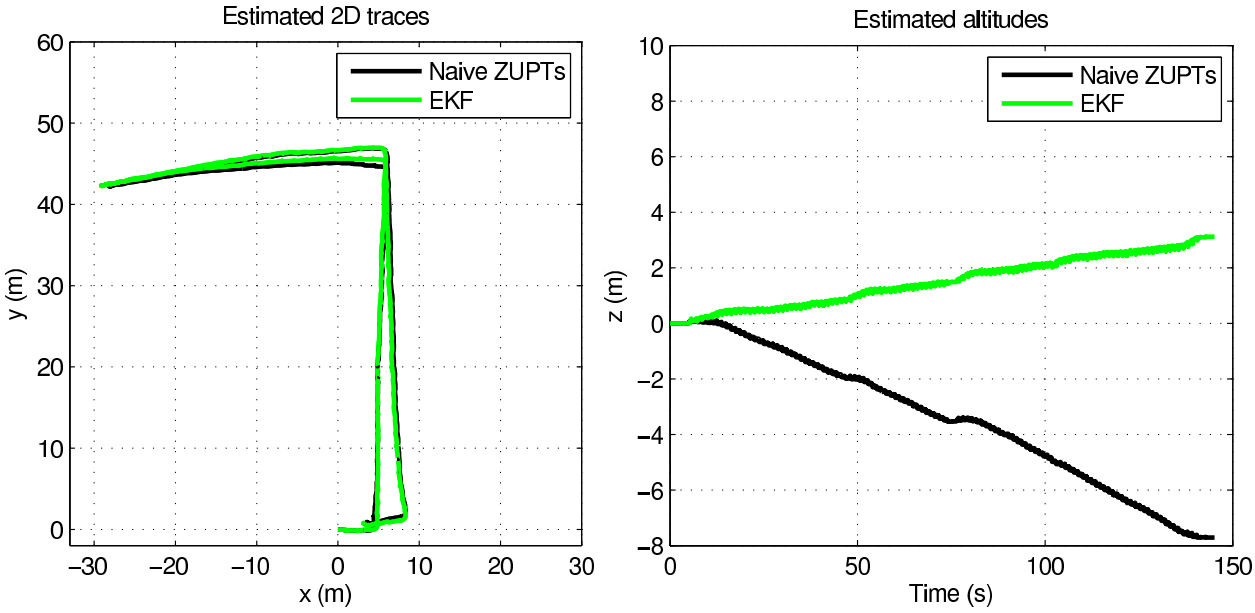

Figure 3.23 Plots of 2D traces and altitude, L2

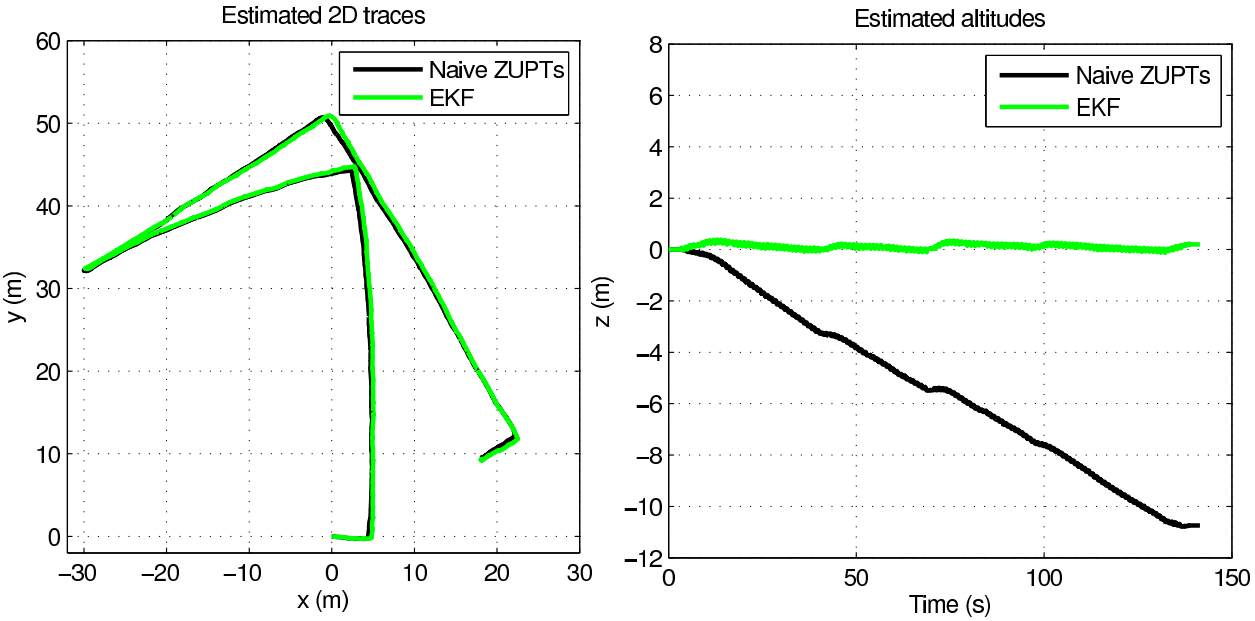

Figure 3.24 Plots of 2D traces and altitude, L3

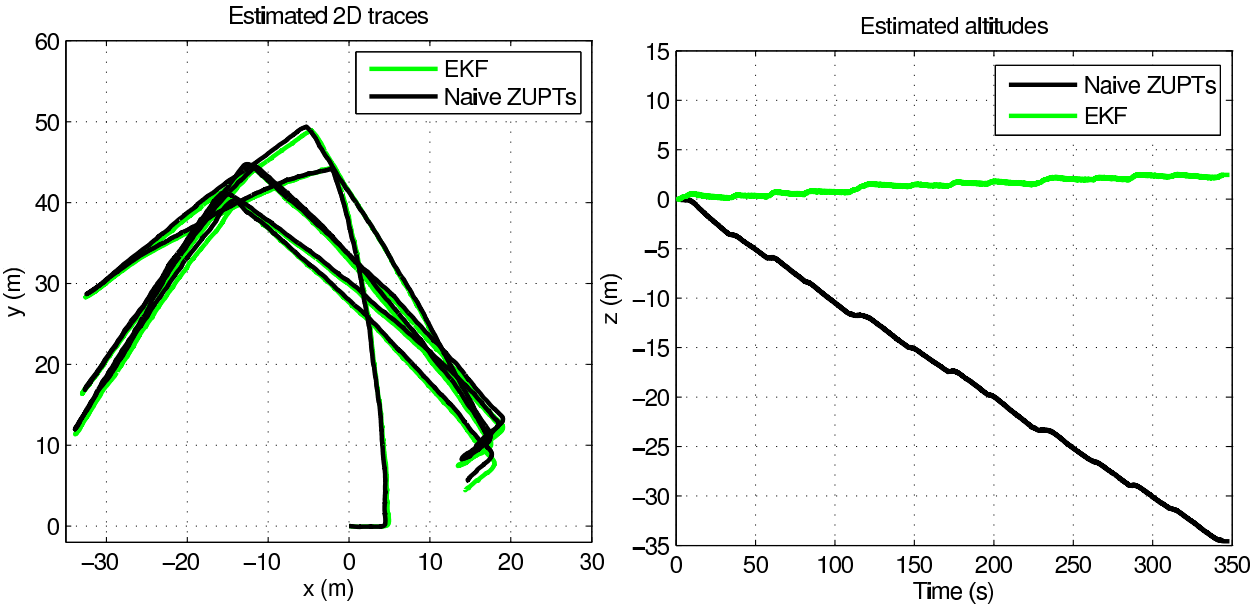

Figure 3.25 Plots of 2D traces and altitude, L4

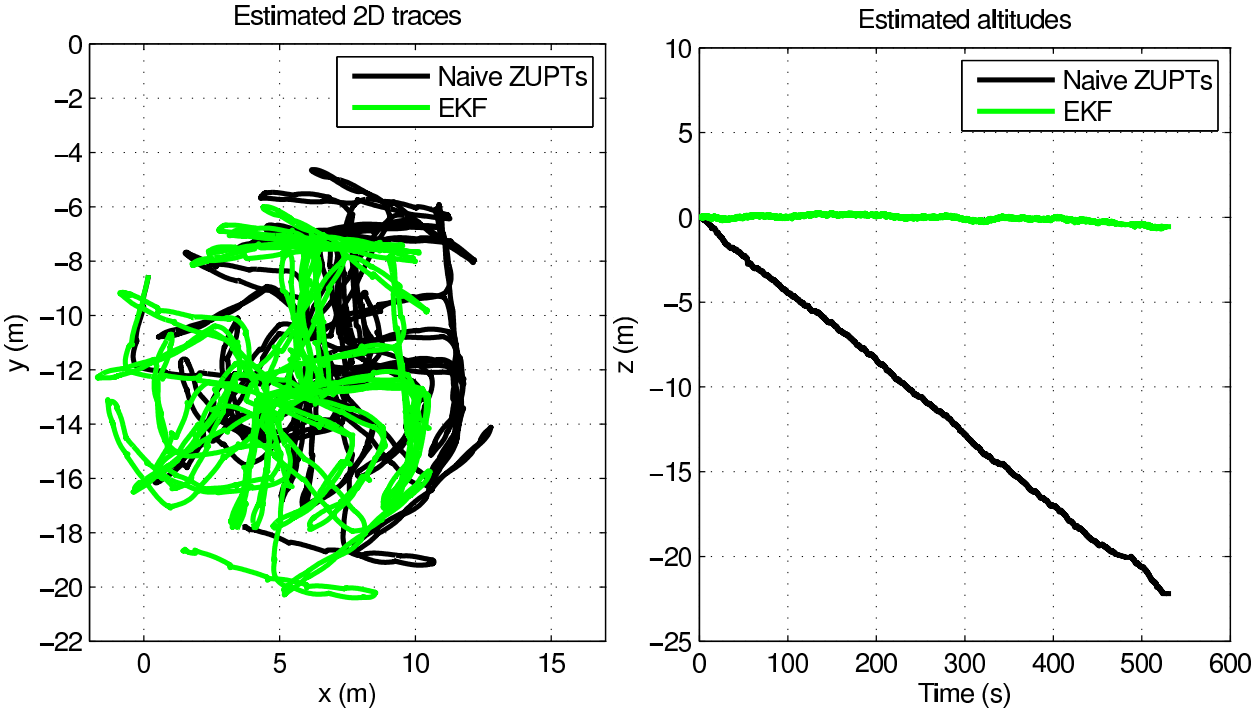

Figure 3.26 Plots of 2D traces and altitude, T1

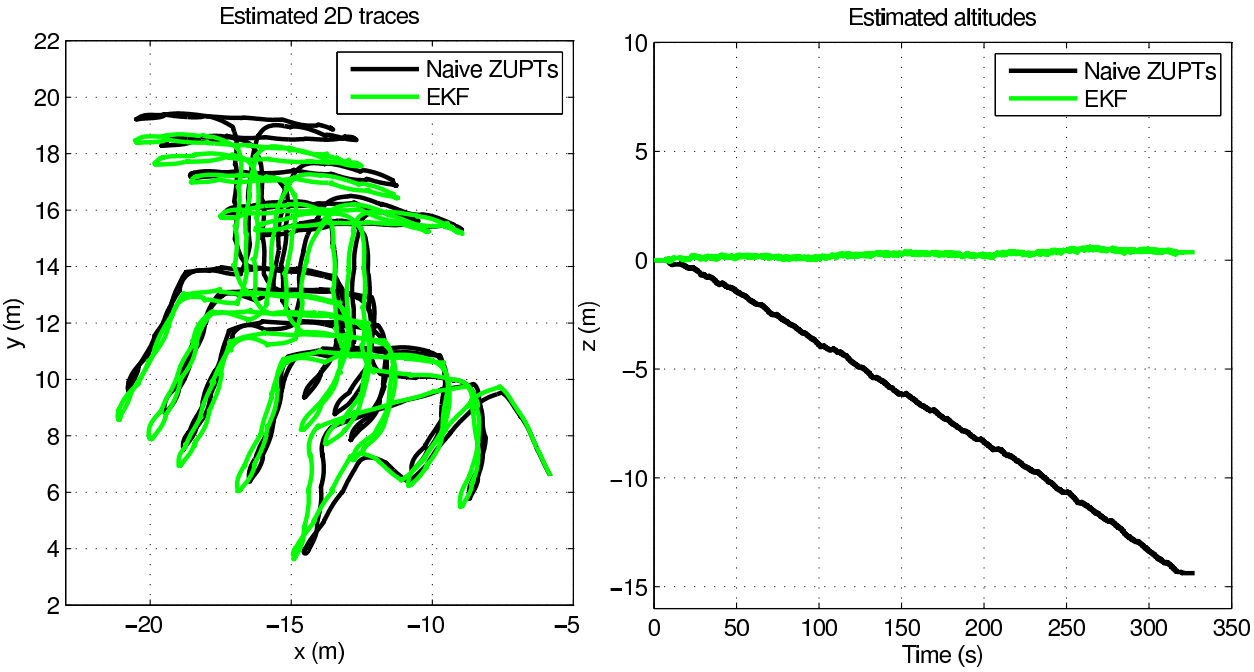

Figure 3.27 Plots of 2D traces and altitude, T2

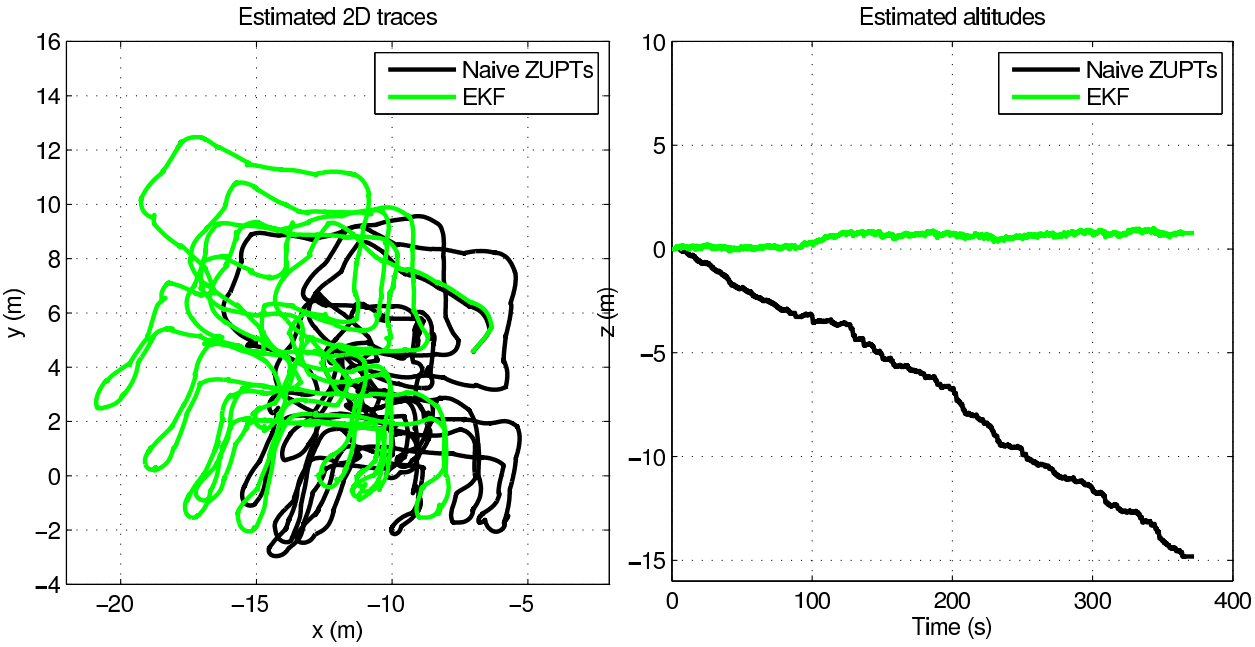

Figure 3.28 Plots of 2D traces and altitude, T3

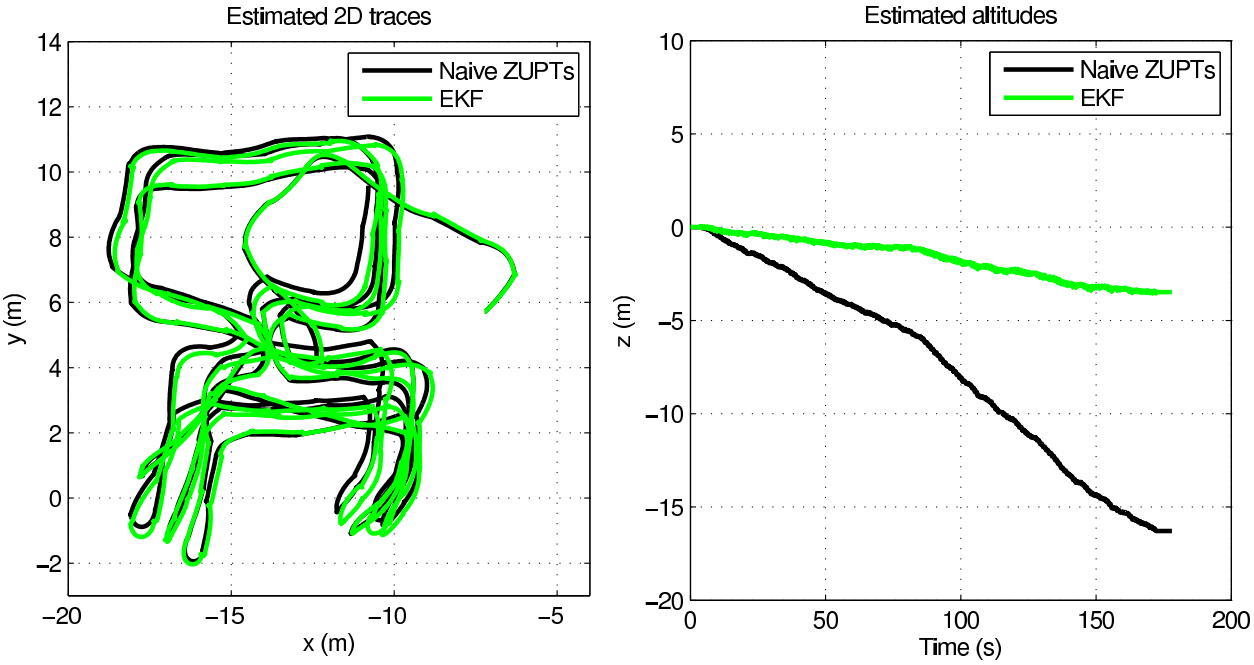

Figure 3.29 Plots of 2D traces and altitude, T4

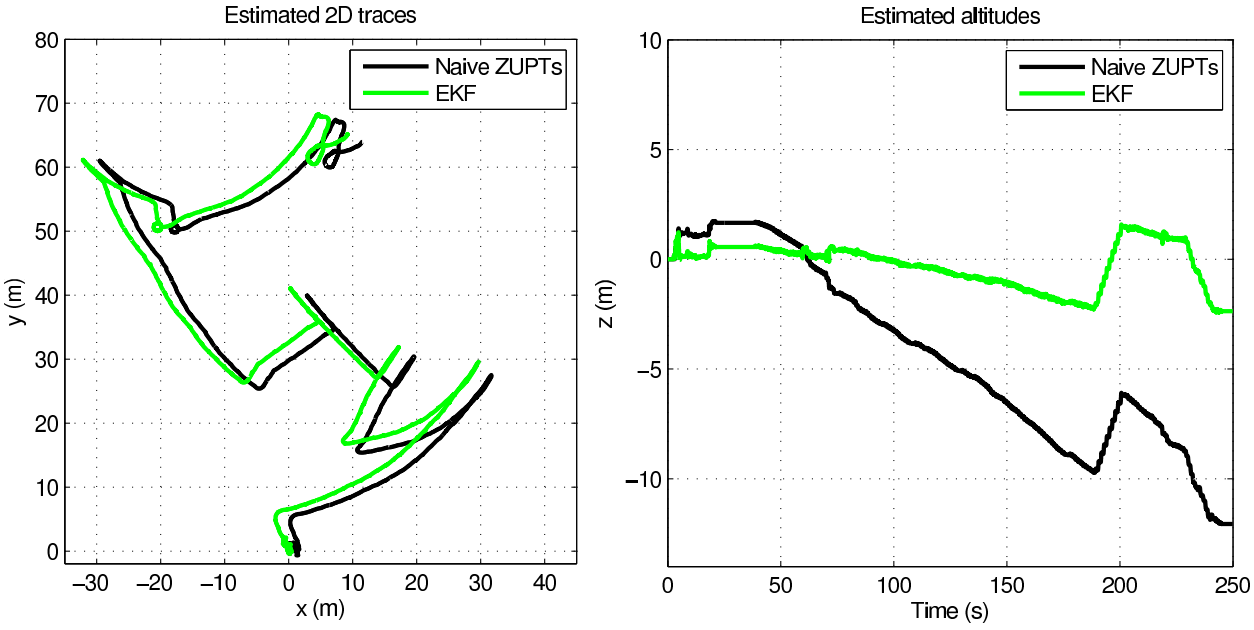

Figure 3.30 Plots of 2D traces and altitude, Biba

Data set	Groundtruth Distances (m)	Naive ZUPTs		EKF	
		Distances (m)	Error (%)	Distances (m)	Error (%)
L1	169.9	129.36	-23.86	176.52	3.9
L2	169.9	170.18	0.16	178.82	5.25
L3	169.9	169.91	0.005	179.33	5.55
L4	508.8	507.16	-0.322	533.55	4.86
T1	665.67*	433.91	n/a	495.35	n/a
T2	399.93*	269.15	n/a	298.13	n/a
T3	379.85*	253.80	n/a	326.03	n/a
T4	211.06*	221.80	n/a	237.79	n/a
Biba	not known	269.22	n/a	301.90	n/a

Table 3.5 Groundtruth distances and distances travelled using the implementations with and without EKF. The starred groundtruth distances were recorded by Ubisense.
(In general, the total distances obtained using the EKF estimates are more than the actual distances travelled. The additional distances are mostly due to the corrections (represented by small zig-zag lines that are almost unobservable in the traces) applied by the EKF during every ZUPT (figure 3.3). It may be possible to obtain better distance estimates by smoothing the EKF estimates.)

Data set	Duration (min)	Naive ZUPTs	EKF
		Altitude value at end of trace $(m)^*$	Altitude value at end of trace $(m)^*$
L1	2.34	15.72	0.53
L2	2.42	7.70	3.12
L3	2.36	10.79	0.20
L4	5.8	34.57	2.46
T1	8.88	22.19	0.55
T2	5.46	14.38	0.36
T3	6.21	14.82	0.77
T4	2.96	16.29	3.48

Table 3.6 The altitude variations for the different data sets (without stairs) using naive ZUPTs and EKF. The starred distances in the table are absolute values. (The altitude traces obtained for almost all the implementations are either continuously decreasing or continuously increasing. Therefore, the (absolute) altitude value at the end of the trace would represent the maximum variation from the true altitude ('0') for each implementation.)

3.6 Filter Tuning

The performance of a Kalman filter relies on the values chosen for the three different covariance matrices: the system noise covariance matrix Q, the measurement noise covariance matrix R and the initial value of the system error covariance matrix $P_{0|0}$. In particular, a delicate adjustment of the Q and R matrices is required to obtain optimal results. Generally, a small Q/R value leads to good tracking but is more noise sensitive and a large Q/R value leads to bad tracking but is less noise sensitive [8, 9, 21].

For all the implementations in this chapter, the following values were used for the Q, R and $P_{0|0}$ matrices.

$$P_{0|0} = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10^{-4}I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{3.39}$$

$$Q = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 10^{-5}I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{3.40}$$

$$R = 0.001I_{3\times3} \tag{3.41}$$

3.7 Summary

A summary of the results of the various implementations in this chapter, all of which use the orientation given by Xsens, is given in this section.

Out of the three algorithms described for stance phase detection, the total distances travelled and altitude variations are greatest for the algorithm using only the 3D angular rate signal (section 3.3.1) for nearly all the data sets. This is because that algorithm identifies only

a single time instance during each stance phase whereas the other two algorithms, threshold-based (section 3.3.2) and HMM-based (section 3.3.3), detect entire stance periods. The threshold-based and the HMM-based stance phase detection algorithms perform equivalently for all the data sets used in this thesis. The threshold-based method is less computationally intensive and can certainly be used for regular walking patterns.

When the orientation is computed by the Xsens, the EKF formulation vastly improves the altitude results but not the orientation or the horizontal path accuracy. This is to be expected however, since the EKF was not specifically designed to improve the orientations. EKFs which do compute the orientation (and do not rely on the Xsens 'black box' algorithm) are formulated in the next chapter.

Chapter 4

EKF tracking using raw IMU measurements

A PDR system has been implemented which extends the implementation described in section 3.2. Instead of using the rotation matrix calculated directly in the Xsens MTx's internal software, the rotation matrix is calculated using the raw IMU measurements (accelerometer and gyroscope data) in this implementation. Additional methods to reduce the attitude and heading errors are also implemented and are described in the subsequent sections.

In the implementation described in section 4.1, zero angular rate updates (ZARUs) are applied in addition to ZUPTs during every stance phase which helps in reducing the heading errors. Section 4.2 proposes a method called 'Heuristic Drift Reduction (HDR)' to reduce heading drift in traces consisting of straight lines. A novel technique where accelerometer data is used to correct orientation is explained in section 4.3.

4.1 ZUPTs and ZARUs

The PDR system implementation described in this section is based on the methods described by Foxlin, Jiménez et al and Sujatha [1, 6, 8]. During stationary periods when the foot is in contact with the ground, the velocities as well as the angular rates should theoretically be zero. Hence, in addition to zero-velocity updates (ZUPTs), zero-angular rate updates (ZARUs) are also applied during every stance phase. This approach also reduces the drift in heading.

The PDR system implementation described in this section is also used for implementing the additional methods explained in the subsequent sections and changes to the implementation are mentioned where appropriate.

The main blocks in this implementation, the strapdown INS mechanization process and the Extended Kalman Filter (EKF), are described below. The outputs of the EKF are used in the strapdown INS equations.

4.1.1 Strapdown INS mechanization process

This section is similar to section 3.2.1 but there are a few additional steps and equations.

The 15-element EKF error state vector at time *k+1* after a ZUPT and ZARU, $\delta x_{k+1} = \begin{pmatrix} \delta\varphi_{k+1}^T & \delta\omega_{k+1}^T & \delta r_{k+1}^T & \delta v_{k+1}^T & \delta a_{k+1}^{b^T} \end{pmatrix}^T$, where, $\delta\varphi_{k+1}^T$ represents the estimated attitude errors, $\delta\omega_{k+1}^T$ represents the estimated gyro biases, δr_{k+1}^T and δv_{k+1}^T denote the estimated position and velocity errors, respectively, and $\delta a_{k+1}^{b^T}$ represents the estimated accelerometer biases, are used to correct the INS outputs. Each of these 3 components has 3 elements each, corresponding to the estimates in the x, y and z axes, respectively.

The INS mechanization process, shown in figure 4.1, consists of the following six steps performed during each time interval.

(i) Gyro and accelerometer biases are compensated using the EKF gyro and accelerometer

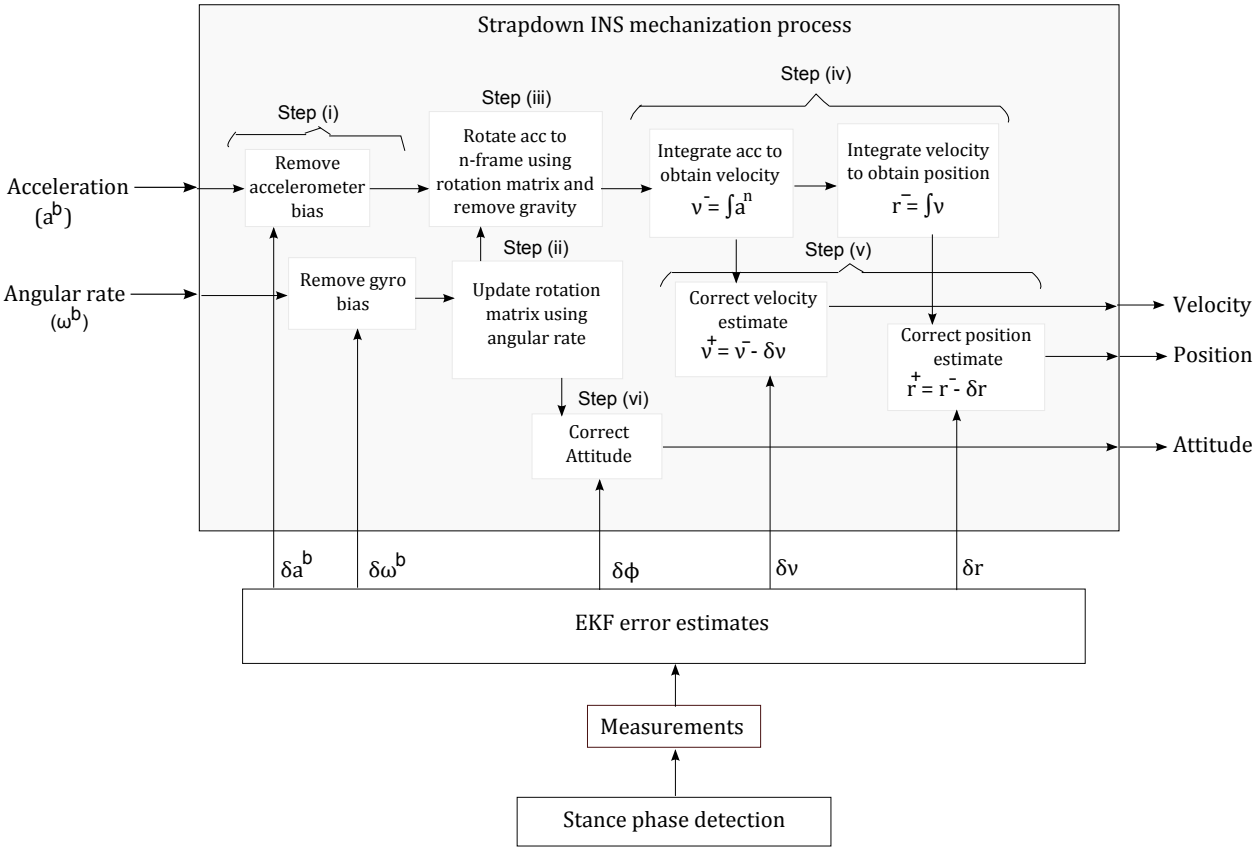

Figure 4.1 Strapdown INS mechanization process

bias estimates (equations 4.1-4.4). The components 4-6 and 13-15 of the error state vector, δx, contain the gyro and accelerometer biases, respectively.

$$\delta\omega_k^b = \delta x_k(4:6) \tag{4.1}$$

$$\delta a_k^b = \delta x_k(13:15) \tag{4.2}$$

$$\omega_{k+1}^{b'} = \omega_{k+1}^b - \delta\omega_k^b \tag{4.3}$$

$$a_{k+1}^{b'} = a_{k+1}^b - \delta a_k^b \tag{4.4}$$

where $\delta\omega_k^b$ and δa_k^b denote the gyro and the accelerometer biases, respectively and $\omega_{k+1}^{b'}$ and $a_{k+1}^{b'}$ denote the bias-compensated gyro and acceleration in the sensor frame.

(ii) The bias-compensated angular rate obtained in the previous step, $\omega_{k+1}^{b'}$, is used to update the attitude (equation 4.5) [6] .

$$C_{b_{k+1|k}}^n = C_{b_{k|k}}^n . \frac{2I_{3\times3} + \delta\Omega_{k+1}\Delta t}{2I_{3\times3} - \delta\Omega_{k+1}\Delta t} \tag{4.5}$$

where $C_{b_{k+1|k}}^n$ denotes the rotation matrix updated using the bias-compensated angular rate measurements at time *k+1* but not yet corrected by the EKF. This rotation matrix is used to transform the acceleration in the sensor frame to the n-frame in the next step. $\delta\Omega_{k+1}$ denotes the skew-symmetric matrix for angular rates used to define the small angular increments in orientation and is given by equation 4.6,

$$\delta\Omega_{k+1} = \begin{pmatrix} 0 & -\omega_{z_{k+1}}^{b'} & \omega_{y_{k+1}}^{b'} \\ \omega_{z_{k+1}}^{b'} & 0 & -\omega_{x_{k+1}}^{b'} \\ -\omega_{y_{k+1}}^{b'} & -\omega_{x_{k+1}}^{b'} & 0 \end{pmatrix} \tag{4.6}$$

(iii) Acceleration in the sensor frame is rotated to the n-frame using the rotation matrix obtained in the previous step and the gravitational component is removed from the z-acceleration in the n-frame (equation 4.7).

$$a^n_{k+1} = C^n_{b_{k+1|k}}.a^{b'}_{k+1} - \begin{pmatrix} 0 & 0 & g \end{pmatrix}^T \tag{4.7}$$

where $C^n_{b_{k+1|k}}$ denotes the rotation matrix calculated in the previous step, a^n_{k+1} denotes the acceleration in the n-frame and 'g' denotes gravity, 9.8 ms^{-2}.

(iv) The acceleration in the n-frame obtained in the previous step is integrated to obtain the velocity in the n-frame (equation 4.8). This velocity estimate is then integrated to obtain the position in the n-frame (equation 4.9).

$$v_{k+1|k} = v_{k|k} + (a^n_k + a^n_{k+1})\frac{\Delta t}{2} \tag{4.8}$$

$$r_{k+1|k} = r_{k|k} + (v_k + v_{k+1})\frac{\Delta t}{2} \tag{4.9}$$

(v) Velocity and position are corrected based on the EKF error estimates (equations 4.10-4.13). The components 7-9 and 10-12 of the error state vector, δx, contain the estimated position and velocity errors, respectively.

$$\delta r_{k+1} = \delta x_{k+1}(7:9) \tag{4.10}$$

$$\delta v_{k+1} = \delta x_{k+1}(10:12) \tag{4.11}$$

$$v_{k+1|k+1} = v_{k+1|k} - \delta v_{k+1} \tag{4.12}$$

$$r_{k+1|k+1} = r_{k+1|k} - \delta r_{k+1} \tag{4.13}$$

(vi) Attitude is corrected based on the EKF attitude (roll, pitch and yaw) error estimates (equation 4.14) [6].

$$C^n_{b_{k+1|k+1}} = C^n_{b_{k+1|k}} . \frac{2I_{3\times3} + \delta\Theta_{k+1}\Delta t}{2I_{3\times3} - \delta\Theta_{k+1}\Delta t} \tag{4.14}$$

where $C^n_{b_{k+1|k+1}}$ denotes the rotation matrix corrected using the attitude errors estimated by the EKF at time *k+1*. $\delta\Theta_{k+1}$ denotes the skew-symmetric matrix for small angles and is given by equation 4.15,

$$\delta\Theta_{k+1} = - \begin{pmatrix} 0 & -\varphi^b_{z_{k+1}} & \varphi^b_{y_{k+1}} \\ \varphi^b_{z_{k+1}} & 0 & -\varphi^b_{x_{k+1}} \\ -\varphi^b_{y_{k+1}} & -\varphi^b_{x_{k+1}} & 0 \end{pmatrix} \tag{4.15}$$

It should be noted that the EKF estimates the errors and the bias only after a stance phase has been detected since zero-velocity and zero-angular rate measurements are available only during stationary periods. After each measurement update, the EKF transfers the error states to the INS and resets the error state vector to zero because those errors are already compensated and incorporated into the INS estimations. Hence, the steps (i), (v) and (vi) are meaningful only after a ZUPT/ZARU.

4.1.2 The Extended Kalman Filter

The EKF error state vector at time *k+1* is given by equation 4.16,

$$\delta x_{k+1} = \begin{pmatrix} \delta\varphi^T_{k+1} & \delta\omega^T_{k+1} & \delta r^T_{k+1} & \delta v^T_{k+1} & \delta a^{b^T}_{k+1} \end{pmatrix}^T \tag{4.16}$$

The state-transition matrix, a 15×15 matrix, at time *k+1* given by equation 4.17,

$$\Phi_{k+1} = \begin{pmatrix} I_{3\times3} & \Delta t C^n_{b_{k+1|k}} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & \Delta t I_{3\times3} & 0_{3\times3} \\ -\Delta t S(a^n_{k+1}) & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & \Delta t C^n_{b_{k+1|k}} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{pmatrix} \tag{4.17}$$

where $S(a^n_{k+1})$ is the skew-symmetric matrix formed from the accelerations in the n-frame given by equation 4.18,

$$S(a^n_{k+1}) = \begin{pmatrix} 0 & -a^n_{z_{k+1}} & a^n_{y_{k+1}} \\ a^n_{z_{k+1}} & 0 & -a^n_{x_{k+1}} \\ -a^n_{y_{k+1}} & a^n_{x_{k+1}} & 0 \end{pmatrix} \tag{4.18}$$

This term causes the attitude errors to propagate into the linear velocity errors in the n-frame. This lets the EKF build up the appropriate correlations in the P matrix so that it also estimates attitude errors using the measurements obtained during stance phases. (see section 3.1).

During every time step, the error covariance matrix, P, is propagated using

$$P_{k+1|k} = \Phi_{k+1} P_{k|k} \Phi^T_{k+1} + Q \tag{4.19}$$

where Φ_{k+1} is the above defined state-transition matrix at time *k+1*, $P_{k+1|k}$ is the error covariance matrix at time *k+1* based on measurements received through k and Q is the system noise covariance matrix.

It should be noted that, although the EKF estimates the errors and the bias only after a ZUPT (and ZARU) since zero-velocity and zero-angular rate measurements are available only during stance phase, the error covariance propagation is performed during each time interval but the error covariance matrix is updated only during a ZUPT (and ZARU) [5].

The error state prediction step, $\delta x_{k+1|k} = \Phi_{k+1}\delta x_k$, need not be implemented since the EKF resets the state vector to zero after each measurement update since the errors are already compensated in the INS estimations.

We assume a virtual measurement $\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T$ of both the INS velocity and angular rate during the stance phase. The difference between the INS velocity and the zero-velocity measurement and the difference between the INS angular rate and the zero-angular rate measurement are input to the EKF. Hence, the measurement vector z is given by

$$z = \begin{pmatrix} z1 \\ z2 \end{pmatrix} \tag{4.20}$$

where 'z1' and 'z2' are given by,

$$z1 = -\begin{pmatrix} \omega_x^{'b} & \omega_y^{'b} & \omega_z^{'b} \end{pmatrix}^T \tag{4.21}$$

$$z2 = -\begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T \tag{4.22}$$

The measurement matrix H, a 6×15 matrix, is given by,

$$H = \begin{pmatrix} 0_{3\times3} & -I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -I_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.23}$$

The set of equations, 4.21, 4.22 and 4.23, can be replaced by the set of equations, 4.24, 4.25 and 4.26.

$$z1 = \begin{pmatrix} \omega_x^{'b} & \omega_y^{'b} & \omega_z^{'b} \end{pmatrix}^T \tag{4.24}$$

$$z2 = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T \tag{4.25}$$

$$H = \begin{pmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.26}$$

The Kalman gain, K, is calculated as follows,

$$K = P_{k+1|k}H^T(HP_{k+1|k}H^T + R)^{-1} \tag{4.27}$$

where R is the measurement noise covariance matrix.

After the EKF receives each measurement, the state vector is updated as follows,

$$\delta x_{k+1} = Kz; \tag{4.28}$$

The error covariance P is then updated using the Joseph form,

$$P_{k+1|k+1} = (I - KH)P_{k+1|k}(I - KH)^T + KRK^T \tag{4.29}$$

4.2 Heuristic Drift Reduction (HDR)

This technique, originally proposed by Borenstein et al [22], is implemented using the guidelines of Jiménez et al [6]. This method is designed to reduce the heading drift by exploiting the fact that walking paths for many applications are straight. Generally, when a person is walking in a straight line, there is not much variation in the yaw angle. Therefore, straight walks can be detected by analyzing the changes in the yaw angle. If the variations in the yaw angle among successive steps remain below a certain threshold, then it may be assumed that the person is walking in a straight line and the heading errors can be reduced by introducing the yaw angle variation as a measurement into the EKF.

Yaw angle at time k is calculated using the rotation matrix:

$$\psi_k = \arctan\left(C^n_{b_{k|k}}(2,1)/C^n_{b_{k|k}}(1,1)\right) \tag{4.30}$$

where $C^n_{b_{k|k-1}}$ denotes the rotation matrix at time k which is not yet corrected by the EKF. A simple implementation of the above technique is described in this section. Let $T_1, T_2, \cdots$ denote the time instances when a step is detected (shown as red points in figure 3.7). The differences between these time instances will generally not be exactly equal and each step will consist of different number of discrete sampling times. For example, the first step (T_1) may be detected after 200 sampling times , the next step (T_2) may be detected after 220 sampling times after the first step and so on. But for many walking gaits, the differences between these time instances will be almost equal and also, each of the steps will exhibit similar patterns. For example, the time instance $T_n + k$ during a particular stance phase would correspond to the time instances $T_{n-1} + k$ and $T_{n-2} + k$ in the stance phases of the previously detected steps and so on.

In this implementation, the mean of the two yaw values at time instances within the previous stance phases that are correlative with the time instance of sample k in its own stance phase is considered. The variation in the yaw angle at time $T_n + k$ is $\Delta\psi_{T_n+k}$, where T_n is the time instance of the last step detected and k is the sampling time in the stance phase of the current step, is obtained as follows:

$$\Delta\psi_{T_n+k} = \psi_{T_n+k} - \frac{(\psi_{T_{n-1}+k} + \psi_{T_{n-2}+k})}{2} \tag{4.31}$$

If the variation in the yaw angle among successive steps, $|\Delta\psi_{T_n+k}|$, is below a given threshold ($th_{\Delta\psi} = 0.06\ rads^{-1}$), then it is assumed to be a straight path and the variation is fed into the EKF as a measurement. If $|\Delta\psi_{T_n+k}|$ is greater than the threshold, then it is considered to be a real variation in the trajectory of the person. The measurement fed into the EKF is given by

$$z3 = \begin{cases} \Delta\psi_{T_n+k} & \text{if } |\Delta\psi_{T_n+k}| < th_{\Delta\psi} \\ 0 & \text{otherwise} \end{cases} \tag{4.32}$$

The measurement vector, z, for applying ZUPTs, ZARUs and HDR is a 7×1 matrix given by

$$z = \begin{pmatrix} z3 \\ z1 \\ z2 \end{pmatrix} \tag{4.33}$$

where $z1$, $z2$ and $z3$ are given by equations 4.24, 4.25 and 4.32, respectively.

The measurement matrix, H, for applying ZUPTs, ZARUs and HDR is a 7×15 matrix given by

$$H = \begin{pmatrix} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{pmatrix} \tag{4.34}$$

In this thesis, this technique was only applied to the 'Long corridor' data sets since those traces consist of long straight paths and were recorded for normal walking.

4.3 Orientation Correction using Accelerometer signal

When the sensor is in semi-static and static states, the magnitude of the acceleration is relatively small and can be neglected with respect to gravity. In this case, the 3D accelerometer unit can be used as an inclinometer. It measures the angle of the sensor unit with respect to gravity. It does not give information about the rotation around the vertical (yaw/heading) but it can be used to estimate the roll and pitch angles. This method will be less accurate for movements with relatively large accelerations [23–25].

This concept has been used partially in the implementation described in section 4.1. The $S(a^n_{k+1})$ term in the state transition matrix (equations 4.17 and 4.18) lets the EKF estimate the roll and pitch errors because these errors are observable from the n-frame accelerations during stationary periods. Here, accelerations in the n-frame are used because the correlations between the n-frame velocities (which depends on n-frame accelerations) and attitude errors are considered.

The method described in this section explicitly uses the inclination estimate obtained from the accelerometer signal (in the sensor frame) during stance phases to correct orientation. The inclination at time k can be calculated by dividing the bias-compensated sensor-frame accelerations by gravity (g=9.8 ms^{-2}) given by

$$\gamma_{acc} = \acute{a^b_k}/g \tag{4.35}$$

The inclination can also be obtained from the gyro signals, given by the third row of the rotation matrix:

$$\gamma_{gyr} = C^n_{b_{k|k-1}}(3,:) \tag{4.36}$$

The difference between γ_{gyr} and γ_{acc} is fed as a measurement into the EKF. The measurement 'z4' is given by equation 4.37,

$$z4 = \gamma^T_{gyr} - \gamma_{acc} \tag{4.37}$$

Note that γ_{gyr} is a row matrix.

The measurement vector, z, combining this implementation with ZUPTs and ZARUs is a 9×1 matrix given by

$$z = \begin{pmatrix} z1 \\ z2 \\ z4 \end{pmatrix} \tag{4.38}$$

where 'z1', 'z2' and 'z4' are given by equations 4.24, 4.25 and 4.38, respectively.

Since the inclination measurement, 'z4' is a non-linear function of the states (attitude and accelerometer bias) of the error state vector, δx, the measurement matrix, H, combining this implementation with ZUPTs and ZARUs becomes a 9×15 Jacobian matrix of partial derivatives:

$$H = \begin{pmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ J & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -I_{3\times3}/g \end{pmatrix} \tag{4.39}$$

where 'g' denotes gravity ($9.8\ ms^{-2}$) and 'J' at time k is given by equation 4.40,

$$J = \begin{pmatrix} 0 & -\cos\delta x_k(1) & 0 \\ \cos\delta x_k(1)\cos\delta x_k(2) & -\sin\delta x_k(1)\sin\delta x_k(2) & 0 \\ -\sin\delta x_k(1)\cos\delta x_k(2) & -\cos\delta x_k(1)\sin\delta x_k(2) & 0 \end{pmatrix} \tag{4.40}$$

where $\delta x_k(1)$ and $\delta x_k(2)$ refer to the roll and pitch errors which are the first and second elements of the error state vector δx_k, respectively.

The measurement vector, z, for combining this implementation with ZUPTs, ZARUs and HDR is a 10×1 matrix given by

$$z = \begin{pmatrix} z3 \\ z1 \\ z2 \\ z4 \end{pmatrix} \tag{4.41}$$

where 'z1', 'z2', 'z3' and 'z4' are given by equations 4.24, 4.25, 4.32 and 4.37, respectively.

The measurement matrix, H, for combining this implementation with ZUPTs, ZARUs and HDR is a 10×15 matrix given by equation 4.42,

$$H = \begin{pmatrix} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} & 0_{1\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ J & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -I_{3\times3}/g \end{pmatrix} \tag{4.42}$$

where 'J' is given by equation 4.40.

4.4 Results

The results of the PDR system implementations explained in this chapter are presented in this section. The four different implementations using combinations of the ZUPTs and ZARUs, HDR and inclination methods described in sections 4.1, 4.2 and 4.3, respectively, are summarized in table 4.1. As mentioned earlier, only 'Long corridor' data sets were used to evaluate implementations 2 and 4 since the HDR technique assumes mostly straight paths.

The stance phase detection algorithm described in section 3.3.2 is used for all the implementations in this section.

Implementation 1	ZUPTs and ZARUs
Implementation 2	ZUPTs, ZARUs and HDR
Implementation 3	ZUPTs, ZARUs and inclination
Implementation 4	ZUPTs, ZARUs, HDR and inclination

Table 4.1 Summary of the different implementations explained in this chapter

Plots of the 2D traces and altitude obtained for the different data sets using the implementations 1, 2, 3 and 4 are given below (figures 4.2-4.10) and are illustrated using solid red, green, black and blue lines, respectively.

It can be observed from the plots that implementation 1 does not give very good results. From plots 4.2 and 4.4, it can be seen that the position estimates are nearly accurate in the beginning of the traces but the errors in orientation accumulate over time. In some cases, the obtained traces are almost unrecognizable (plots 4.6-4.9). Hence, it becomes necessary to use other techniques in addition to ZUPTs and ZARUs to further reduce the attitude and heading errors.

For the Twente data sets, the trace obtained for T2 using implementation 3 (figure 4.7) appears to be slightly better than the trace obtained for T1 using implementation 3 (fig-

ure 4.6). It can also be seen from figures 4.8 and 4.9 that the trace obtained for T3 using implementation 3 is almost unrecognizable whereas the trace obtained for T4 using implementation 3 is very good. Similar observations were made between plots 3.17 and 3.18 and plots 3.19 and 3.20, respectively, in the results section in the previous chapter (section 3.4) where the sensor's orientation was used. The differences between the recordings of these data sets are given in section 2.4. From the above observations, it may be assumed that certain factors or combinations of them could have affected the raw IMU measurements (accelerometer and gyroscope data) obtained for T1 and T3. But they are independent of the sensor's quality since the same sensor was used to record T2 and T4. The factors, in general, could include different gaits, users, sampling frequencies, durations of traces, walking speeds, trail topologies, the way the sensor is attached to the foot and temperature. Hence, it becomes important to find out how the sensor can be tuned depending on the environment where it will be used, to provide measurements using which optimal results can be obtained.

By comparing the plots 4.10 and 3.21 obtained for the Biba data set, it can be observed that the trace obtained without using the orientation provided by the Xsens algorithm (plot 4.10) is closer to the actual trace than the trace obtained using the Xsens orientation results (plot 3.21). This could be because the in-built algorithm for orientation calculation was less optimal in the older version of Xsens MTx IMU which was used to record the Biba data set. It is also possible that the high level of magnetic disturbance in the Biba workshop may have affected the magnetometer readings which were used in the orientation calculation. By contrast, none of the implementations in this chapter make use of the magnetometer readings.

By looking at the various plots and considering the above observations and assumptions, it can be said that implementation 3 performs well and produces much better results than implementation 1 for most of the data sets. But, it is hard to compare the traces obtained

for the Biba data set using implementations 1 and 3 (figure 4.10). The trace obtained using implementation 3 is smooth and appears well spread out than the trace obtained using implementation 1. But, some fluctuations are observed in the beginning of the trace with implementation 3 which are also reflected in the altitude plot. Also, the path from the first spiral staircase to the second spiral staircase (and back to the starting point) has drifted from the true path in the implementation 3 trace while it appears to be correct in the trace obtained using implementation 1. It may be possible to correct these errors in the Biba trace and also obtain better results for T1 and T3 using implementation 3 by further tuning the EKF. This is discussed in the next section (section 4.5).

For some of the data sets, the horizontal traces obtained using the Xsens's orientation in chapter 3 are better than those obtained using implementation 3 in this chapter. For example, the traces obtained for T3 shown in figures 3.19 and 4.8.

From the plots in figures 4.2 and 4.4, it can be seen that the HDR technique, used in implementations 2 and 4, performs well and produces good results. The trace obtained for L4 (figure 4.4) using implementation 4 seems to be very accurate and appears almost the same as the surveyed path (figure 2.4).

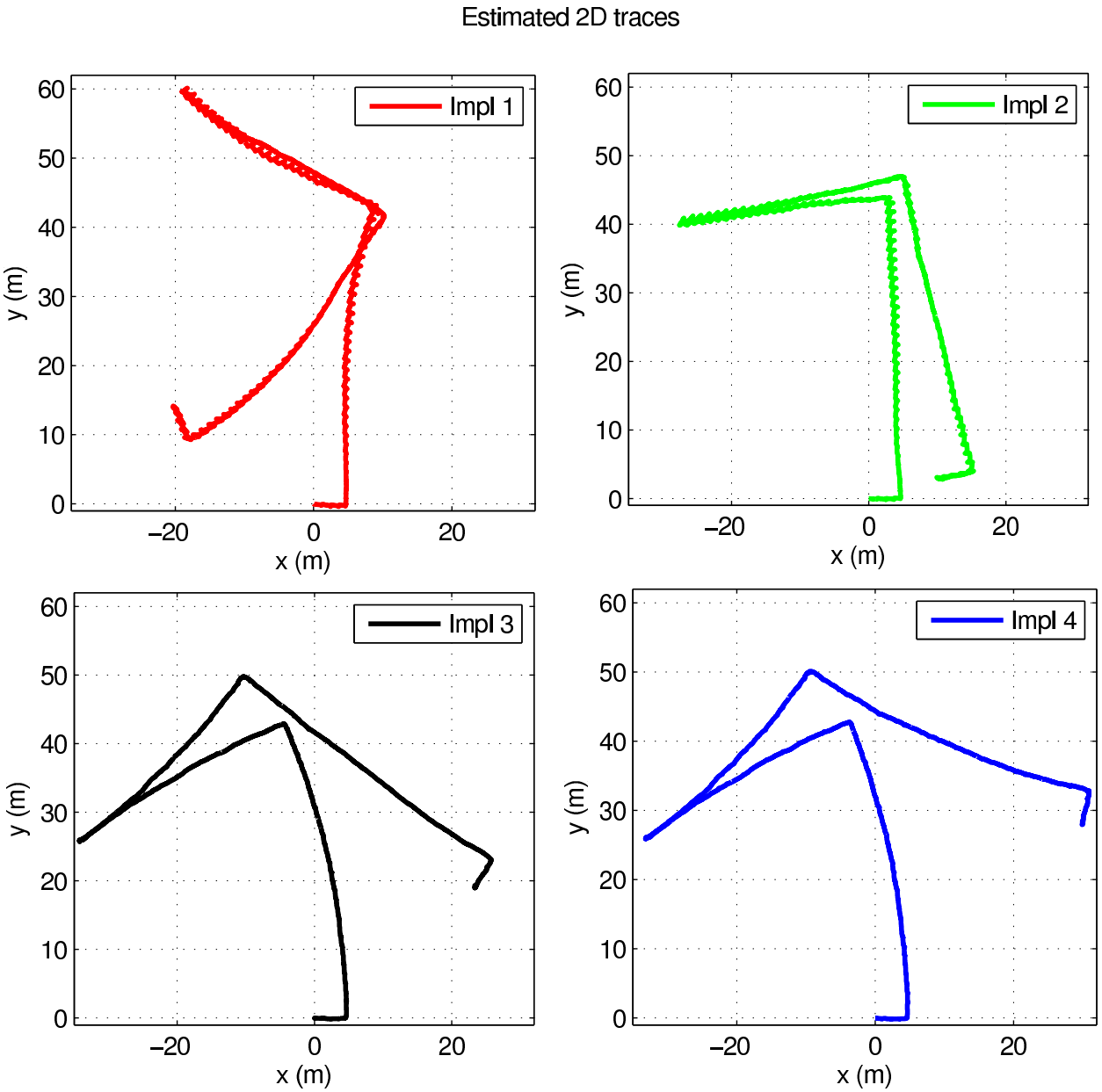

Figure 4.2 Plots of 2D traces, L3

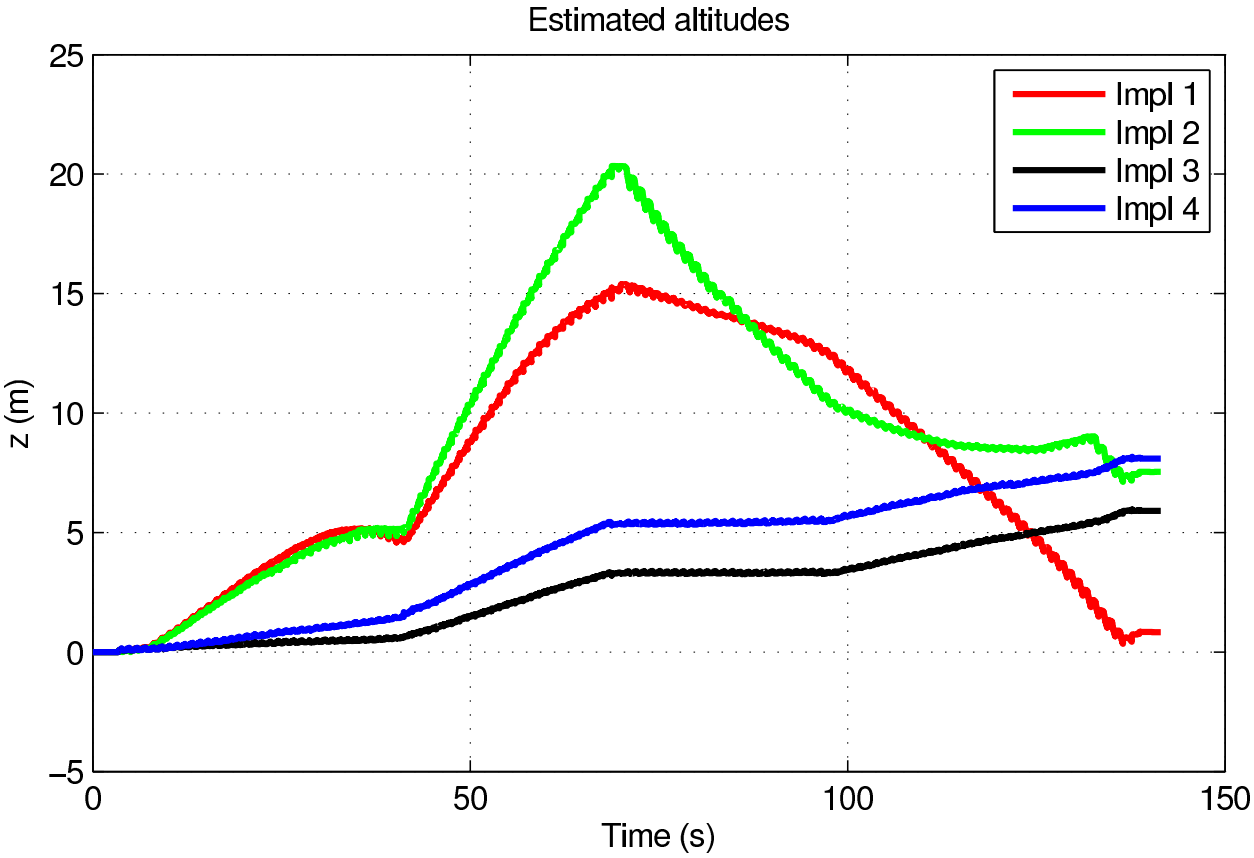

Figure 4.3 Plots of altitudes, L3

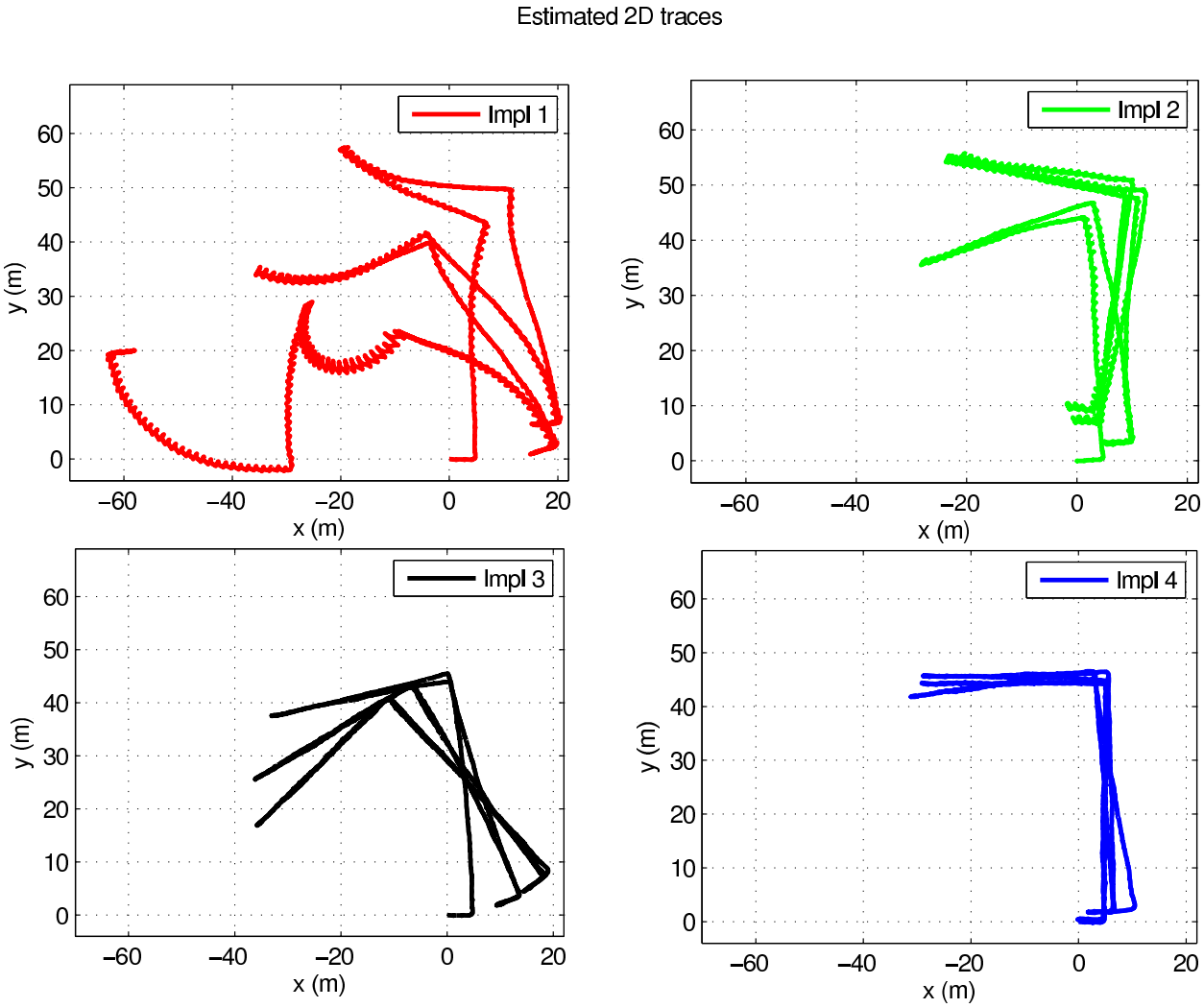

Figure 4.4 Plots of 2D traces, L4

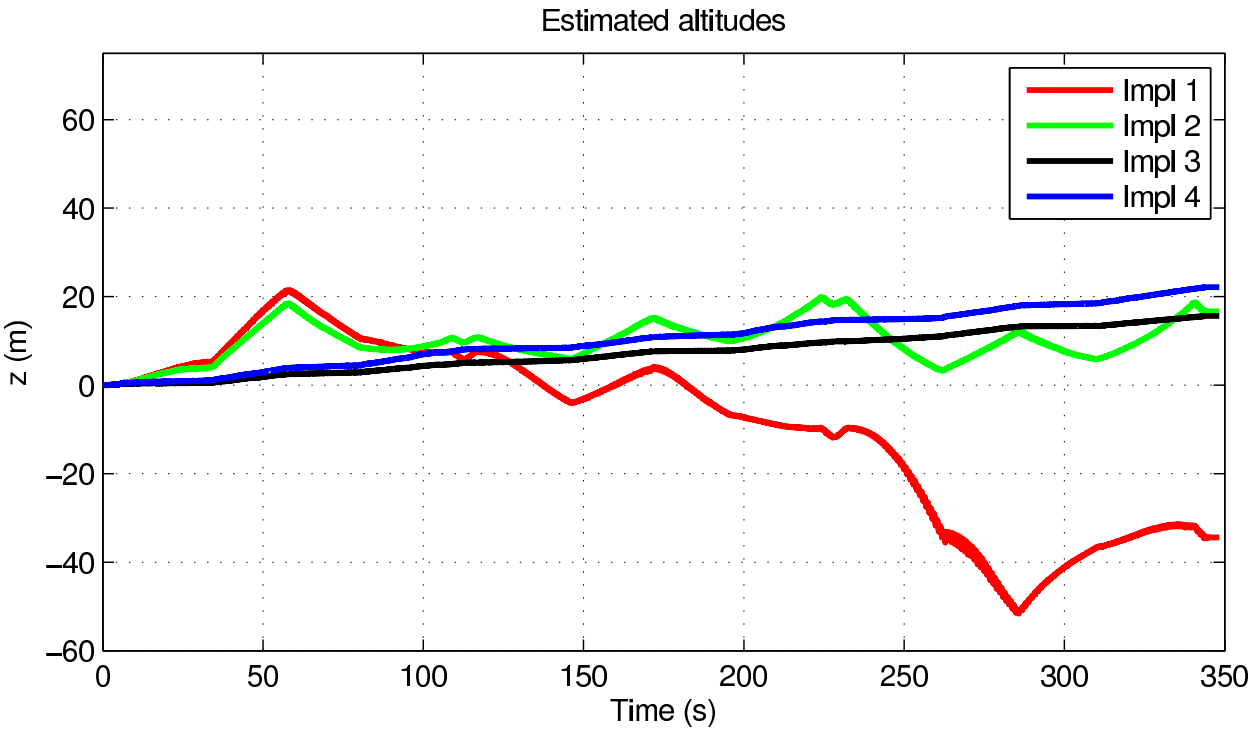

Figure 4.5 Plots of altitudes, L4

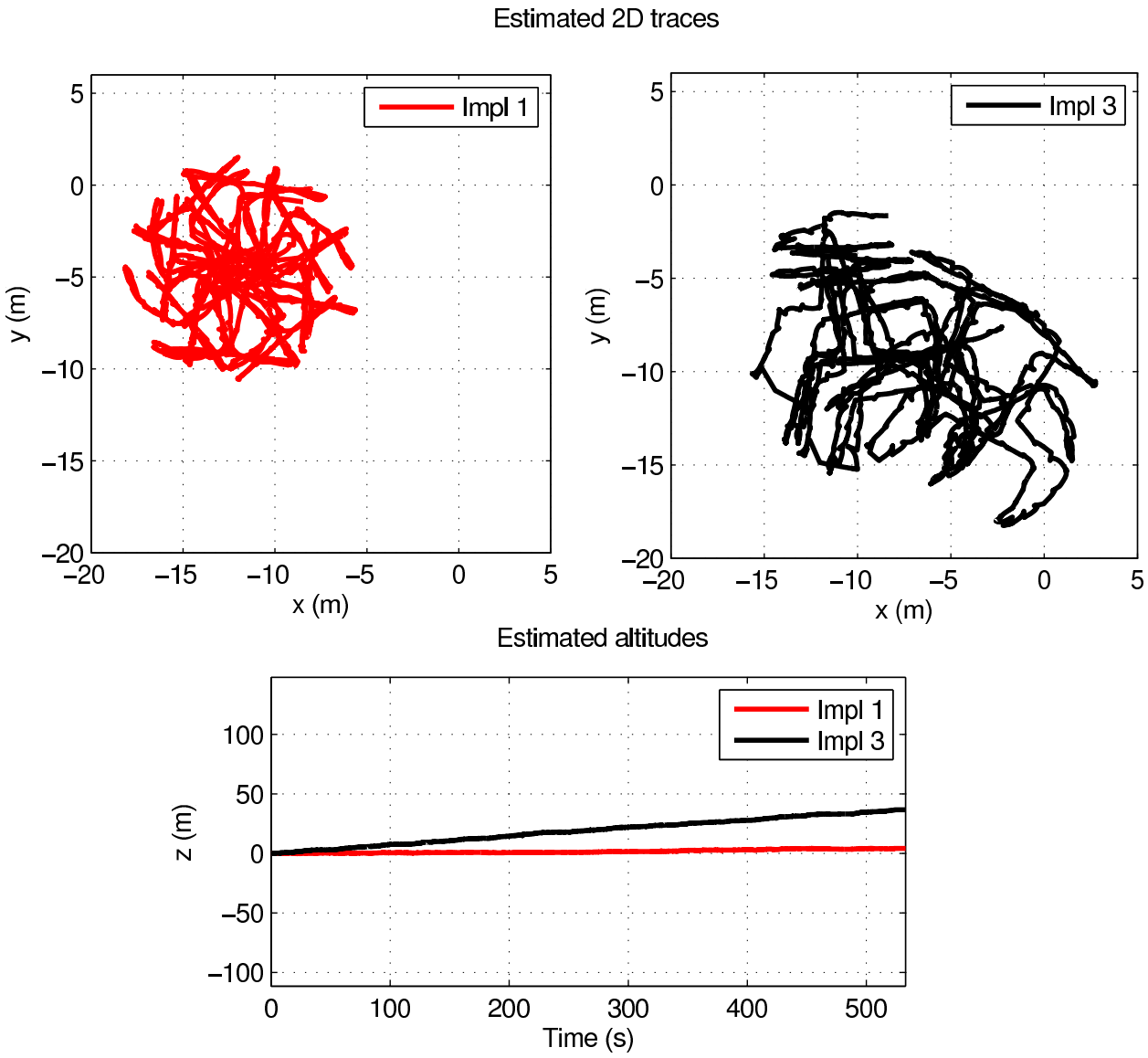

Figure 4.6 Plots of 2D traces and altitudes, T1

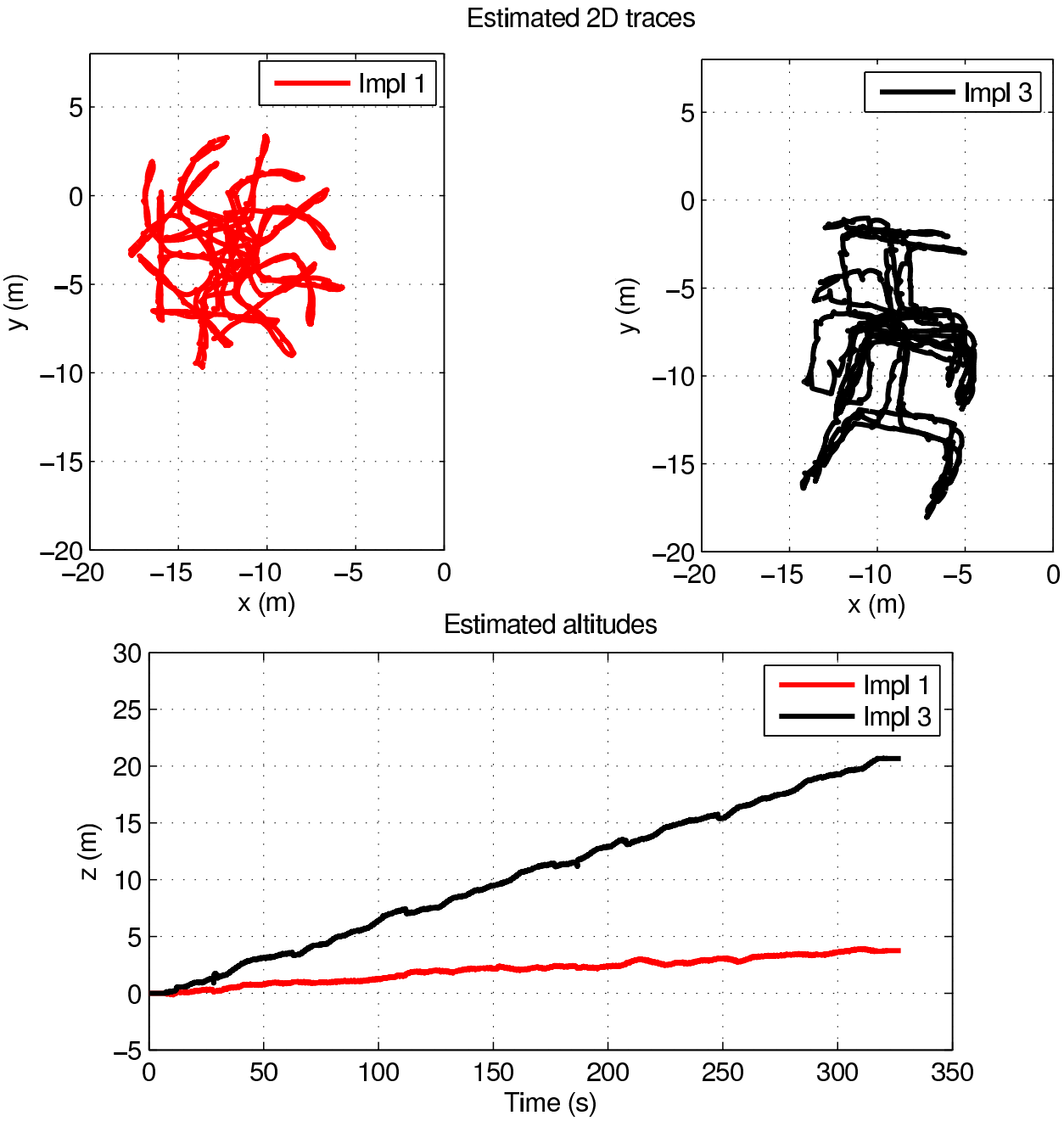

Figure 4.7 Plots of 2D traces and altitudes, T2

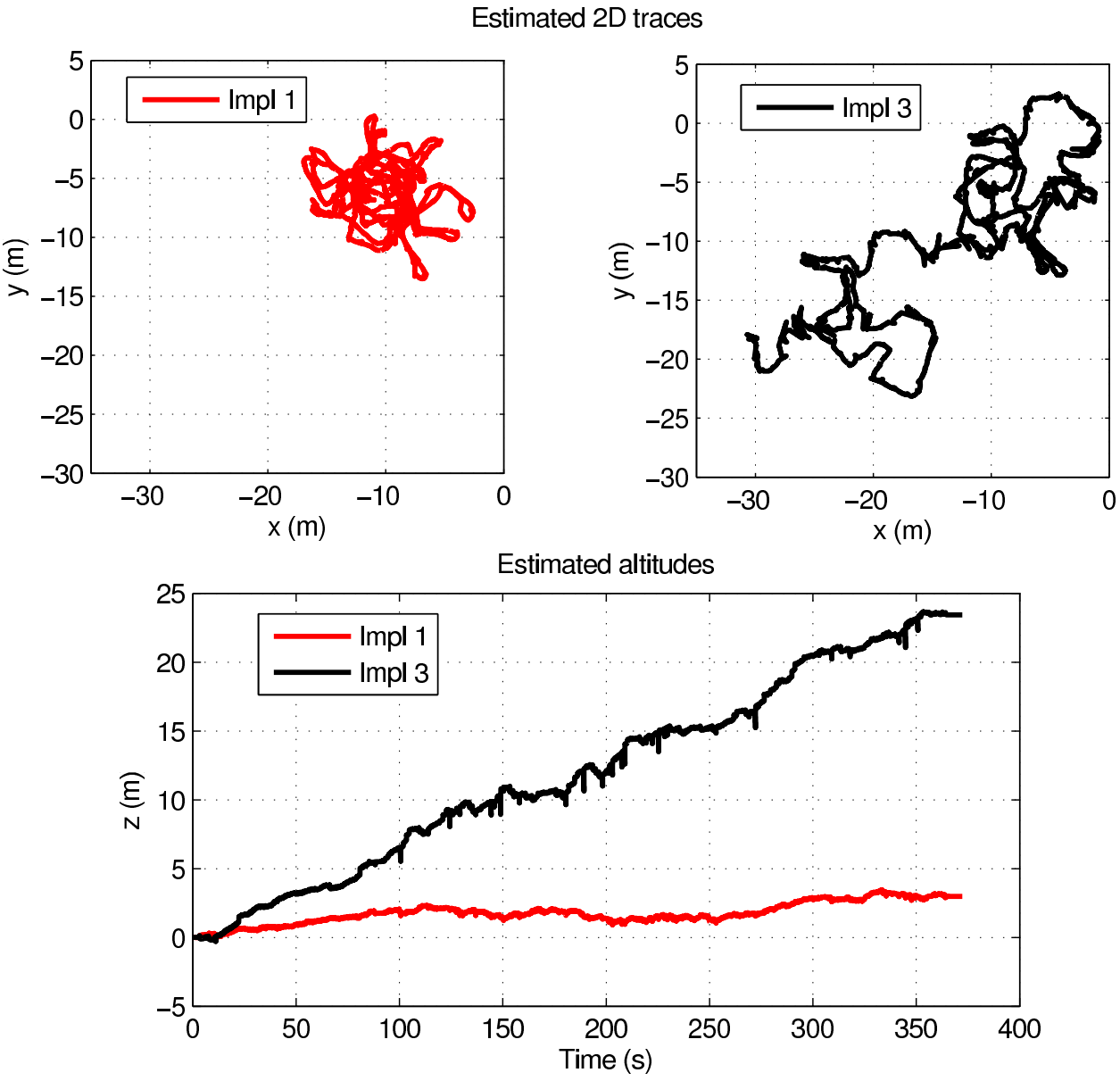

Figure 4.8 Plots of 2D traces and altitudes, T3

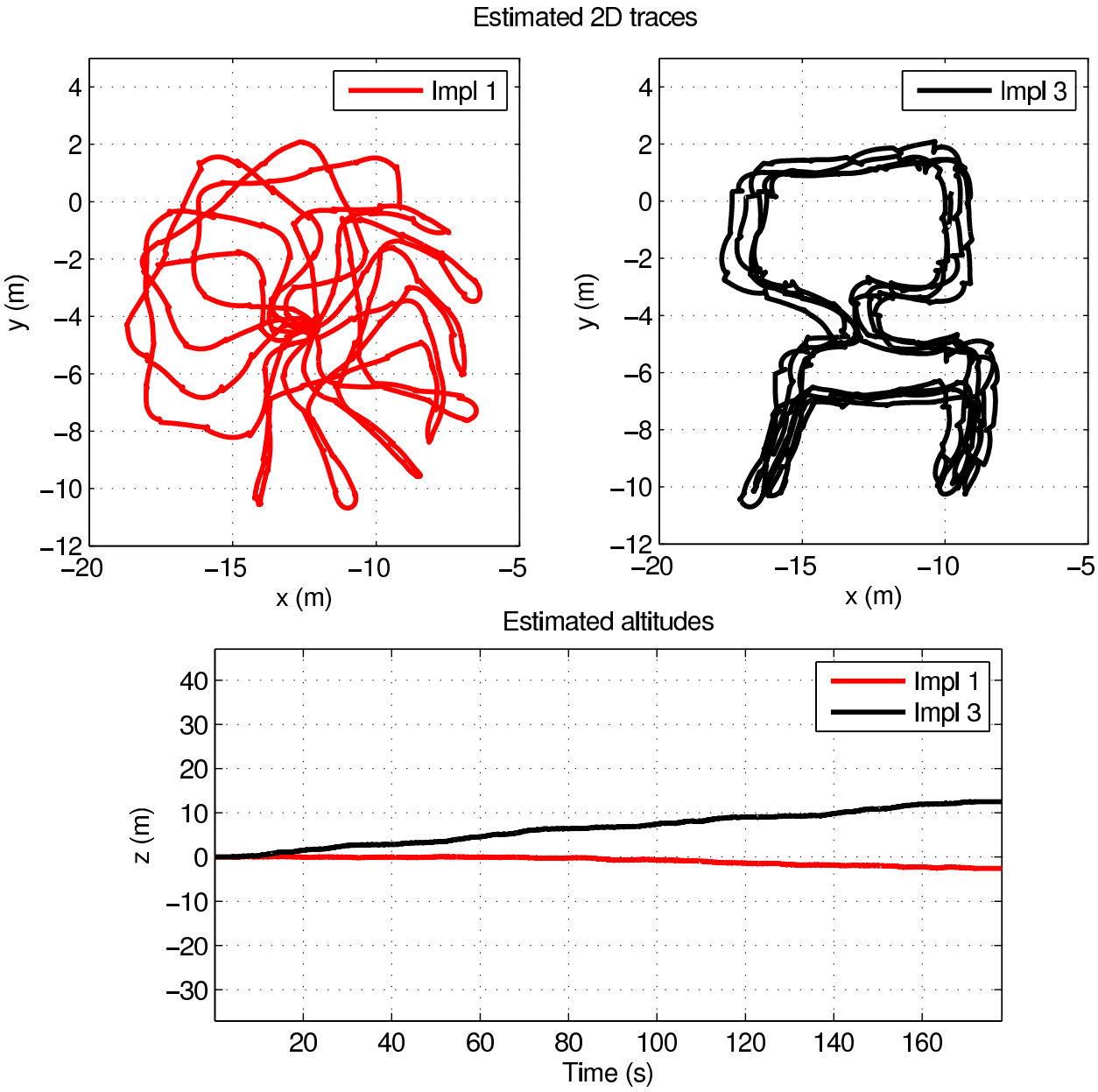

Figure 4.9 Plots of 2D traces and altitudes, T4

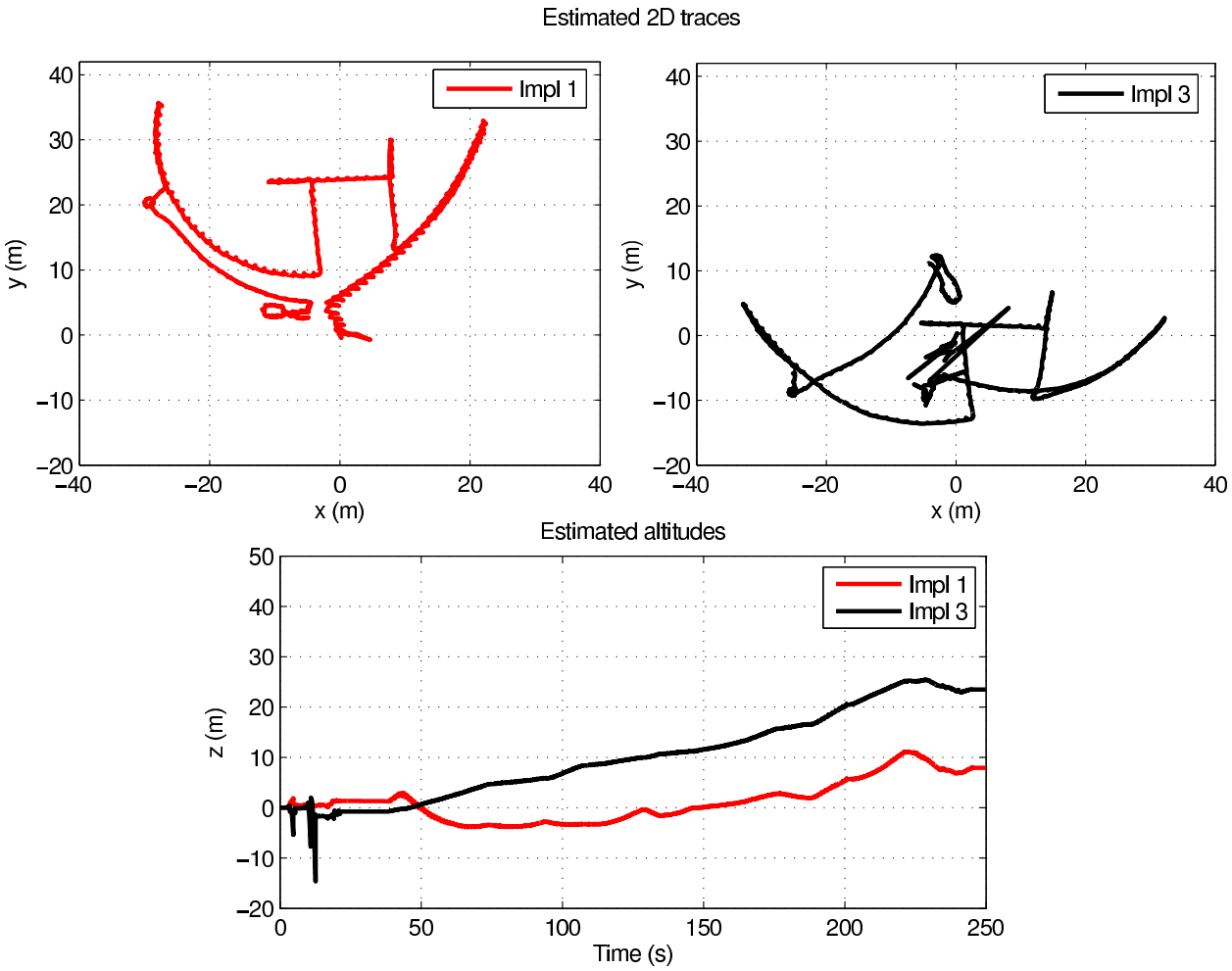

Figure 4.10 Plots of 2D traces and altitudes, Biba

4.5 Filter Tuning

The following values were used for the EKF parameters for the different implementations (implementations 1, 2, 3 and 4) described in this chapter.

In all the implementations, the following value was used for the $P_{0|0}$ matrix:

$$diag(P_{0|0}) = \begin{pmatrix} 10^{-4} & 10^{-4} & 10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-4} & 10^{-4} & 10^{-4} & 0 & 0 & 0 \end{pmatrix}. \tag{4.43}$$

Values used for the Q and R matrices for the different implementations are given in table 4.2.

Implementation	diag(Q)	R
1	$\begin{pmatrix} 10^{-4} & 10^{-4} & 10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-2} & 10^{-2} & 10^{-2} & 0 & 0 & 0 \end{pmatrix}$	$0.001I_{6\times6}$
2	$\begin{pmatrix} 10^{-6} & 10^{-4} & 10^{-6} & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-2} & 10^{-2} & 10^{-2} & 0 & 0 & 0 \end{pmatrix}$	$0.001I_{7\times7}$
3	$\begin{pmatrix} 10^{-6} & 10^{-4} & 10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-2} & 10^{-2} & 10^{-6} & 0 & 0 & 0 \end{pmatrix}$	$0.001I_{9\times9}$
4	$\begin{pmatrix} 10^{-4} & 10^{-3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-2} & 10^{-2} & 10^{-6} & 0 & 0 & 0 \end{pmatrix}$	$0.001I_{10\times10}$

Table 4.2 Q and R matrix values for the different implementations.

As mentioned earlier, the performance of the Kalman filter depends on the values chosen for the three different covariance matrices, particularly the system noise and measurement noise covariance matrices, Q and R, respectively. It is common for these matrices to be diagonal and constant [9]. Hence, the filter is tuned by mostly tweaking the values of the diagonal elements of the Q and R matrices. For the implementations described in chapter 3 where the MTx's given orientation was used, it was not very hard to tune the EKF. The error state vector δx consisted only of 9 elements and small changes in the values assigned to the EKF parameters did not produce any great differences in the results. But, for the implementations presented in this chapter, the error state vector δx consists of 15 elements

and the corresponding Q matrix is a 15×15 matrix and even small changes in the values assigned to these parameters seemed to affect the results greatly. Hence, there is further room for improvement by refining the EKF. This may also help in smoothing the estimates and reducing the altitude variations.

4.6 Summary

A summary of the results of the various implementations in this chapter where orientation is calculated using raw IMU measurements (accelerometer and gyroscope data), is given here.

In most cases, the results obtained using only the ZUPTs and ZARUs are less optimal than the results obtained using the implementation combining ZUPTs, ZARUs and the inclination technique (section 4.3). Therefore, using an inclination measurement (derived from accelerometer readings) to correct the orientation in addition to the algorithms previously proposed [1,6,8] yields significantly better results than using only the algorithms previously proposed [1,6,8]. The HDR method (section 4.2), designed for mostly straight paths, performs well when combined with ZUPTs and ZARUs, and even better results are obtained when the inclination technique is used in addition to these methods.

The proprietary 'black box' algorithm of the Xsens MTx IMU makes use of specialized knowledge of the sensor hardware. It assigns unknown weights to the raw IMU measurements, including magnetometer readings, in the calculation of orientation depending on the sensor output settings. By contrast, the implementations explained in this chapter do not use magnetometer readings. Only accelerometer and gyroscope data are used to calculate the orientation, and some additional methods for reducing heading drift are implemented. It is possible to include magnetometer readings in the calculation of orientation and refine these implementations to obtain better results. From the results presented in this chapter, it can be said that it is possible to obtain good results without using the orientation provided

by the Xsens proprietary algorithm. This allows less expensive sensors to be used, and reliance on the proprietary algorithm can be eliminated. It should also be noted that sensors providing only raw IMU measurements also have higher sampling rates.

Chapter 5

Conclusions

The key research contributions of this thesis are presented in section 5.1. Additional observations and topics for further research are listed in sections 5.2 and 5.3, respectively.

5.1 Contributions

There are three key research contributions in this thesis:

1. The threshold-based stance phase detection [6] and the HMM based stance phase detection [18] work roughly the same for all the data sets used in the thesis. **The threshold based method is less computationally intensive and can be used for regular walking patterns.**

2. The new EKF/ZUPTs formulation implemented which uses the Xsens orientation gives similarly good horizontal path results and **vastly improves the altitude results** compared to tracking using the Xsens orientation and ZUPTs directly.

3. When the orientation is calculated using raw accelerometer and gyroscope readings, using an inclination measurement (derived from accelerometer readings) to correct the

orientation in addition to the algorithms previously proposed [1, 6, 8] **yields significantly better results than using only the algorithms previously proposed** [1, 6, 8].

5.2 Additional Observations

Some of the additional observations in the thesis are listed below:

1. Filter tuning is very important in order to consistently obtain optimal results. If the number of elements in the error state vector increases, the Q matrix becomes larger and it will be harder to tune the EKF.

   For the implementations in this thesis, setting the diagonal elements corresponding to the gyro and accelerometer biases in the Q matrix to zero seemed to yield better results. Also, changing the values of the diagonal elements corresponding to the position error in the Q matrix did not seem to affect the results. It was also observed that changing the values of the first three diagonal elements (corresponding to roll, pitch and yaw angles) of the Q matrix seemed to result in a trade-off between the orientation in the horizontal traces and the altitude variations for the implementations described in chapter 4.

   Certain tuning philosophies could be used to determine suitable values for the Q and R matrices. For example, the $P_{0|0}$ and Q matrices can be fixed and then R can be varied by trial and error or the $P_{0|0}$ and R matrices can be fixed and then Q can be varied by trial and error to find optimal values for these matrices which give stable results [9].

2. It was observed that the contribution of the zero-angular rate updates (ZARUs) was insignificant compared that of the zero-velocity updates (ZUPTs) for the implemen-

tations described in chapter 4. This could be because the angular rates estimated by the INS during stance phases are already close to zero. By contrast, the velocity errors will be much higher because the small errors in the accelerations are integrated into larger errors in the velocity during the velocity estimation in the INS.

5.3 Future work

The important topics considered for future work are listed below:

1. The threshold-based stance phase detection [6] and the HMM based stance phase detection [18] algorithms can be further evaluated using data consisting of irregular walking patterns.

2. As mentioned earlier (section 2.2), before an INS is used to provide a navigation solution, that navigation solution, consisting of position, velocity and attitude, must be initialized. Position and velocity must be initialized using external information but attitude can be initialized either by using external information or by sensing gravity and the Earth's rotation [9].

   The attitude is also initialized in the Xsens's in-built algorithm for attitude computation. For the implementations in chapter 4, where the attitude was calculated using raw accelerometer and gyroscope data, the attitude was initialized using the Xsens's initial attitude solution. But, it is important to initialize the attitude using 'self-alignment' and without using external information.

   Self-alignment is performed when the INS is stationary and the roll and pitch can be initialized with all but the poorest inertial sensors and heading is often initialized using a magnetic compass [9].

3. For data sets recorded in the absence of magnetic interferences, orientation can be improved by using the magnetometer readings obtained from Xsens in addition to the accelerometer and gyroscope data for the calculation of orientation in chapter 4.

Bibliography

[1] Eric Foxlin. Pedestrian Tracking with Shoe-Mounted Inertial Sensors. *IEEE Computer Graphics and Applications*, November/December 2005.

[2] Carl Fischer and Hans Gellersen. Location and Navigation Support for Emergency Responders: A Survey. *Pervasive Computing, Published by the IEEE CS*, pages 38–47, January-March 2010.

[3] Lauro Ojeda and Johann Borenstein. Non-GPS Navigation for Security Personnel and First Responders. *Journal of Navigation*, 60(3):391–407, September 2007.

[4] Stéphane Beauregard. Omnidirectional Pedestrian Navigation for First Responders. *IEEE*, 2007.

[5] W. Todd Faulkner, Robert Alwood, David W. A. Taylor, and Jane Bohlin. Altitude Accuracy While Tracking Pedestrians Using a Boot-mounted IMU. *IEEE*, 2010.

[6] A.R. Jiménez, F. Seco, J.C. Prieto, and J. Guevara. Indoor Pedestrian Navigation using an INS/EKF framework for Yaw Drift Reduction and a Foot-mounted IMU. *WPNC 2010: 7th Workshop on Positioning, Navigation and Communication*, 2010.

[7] Raúl Feliz, Eduardo Zalama, and Jaime Gómez García-Bermejo. Pedestrian tracking using inertial sensors. *Journal of Physical Agents*, 3(1):35–43, January 2009.

[8] Sujatha Rajagopal. Personal Dead Reckoning System with Shoe Mounted Inertial Sensors. Master's thesis, KTH Royal Institute of Technology, Sweden, 2008.

[9] Paul D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. 2008.

[10] Oliver J. Woodman. An introduction to inertial navigation. Technical Report 696, University of Cambridge, August 2007.

[11] Robert Grover Brown and Patrick Y.C. Hwang. *Introduction to random signals and applied Kalman filtering : with MATLAB exercises and solutions* . 3rd edition, 1997.

[12] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical Report 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, July 2006.

[13] Kavitha Muthukrishnan. *Multimodal Localisation: Analysis, Algorithms and Experimental Evaluation.* PhD thesis, University of Twente, September 2009.

[14] Xsens Technologies B.V. *MTi and MTx User Manual and Technical Documentation*, March 2006.

[15] S. Godha, G. Lachapelle, and M. E. Cannon. Integrated GPS/INS System for Pedestrian Navigation in a Signal Degraded Environment. *ION GNSS*, September 2006.

[16] David H. Titterton and John L. Weston. *Strapdown Inertial Navigation Technology.* 2nd edition, 2004.

[17] Eric Foxlin. Intertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter. *IEEE*, 1996.

[18] Young Soo Suh and Sangkyung Park. Pedestrian Inertial Navigation with Gait Phase Detection Assisted Zero Velocity Updating. *IEEE Proceedings of the 4th International Conference on Autonomous Robots and Agents*, pages 336–341, February 2009.

[19] Brian D.O. Anderson. From Wiener to Hidden Markov Models. *IEEE Control Systems*, 19(3):41–51, 1999.

[20] Valérie Renaudin, Okan Yalak, Phillip Tomé, and Bertrand Merminod. Indoor Navigation of Emergency Agents. *European Journal of Navigation*, 5(3):36–45, July 2007.

[21] Adrian Schumacher. Integration of a GPS aided Strapdown Inertial Navigation System for Land Vehicles. Master's thesis, KTH Royal Institute of Technology, Sweden, 2006.

[22] Johann Borenstein, Lauro Ojeda, and Surat Kwanmuang. Heuristic Reduction of Gyro Drift in IMU-based Personnel Tracking Systems. *SPIE Defense, Security and Sensing Conference, Orlando, Florida, USA*, pages 1–11, April 2009.

[23] Ashutosh Saxena, Gaurav Gupta, Vadim Gerasimov, and Sébastien Ourselin. In Use Parameter Estimation of Inertial Sensors by Detecting Multilevel Quasi-static States. *Ninth International Conference on Knowledge-Based Intelligent Information and Engineering Systems(KES'05), LNCS, Melbourne, Australia*, 3684:595–601, August 2005.

[24] H. J. Luinge and P. H. Veltink. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and Computing*, 43:273–282, 2005.

[25] J. Vaganay, M. J. Aldon, and A. Fournier. Mobile Robot Attitude Estimation by Fusion of Inertial Data. *IEEE International Conference on Robotics and Automation*, 1:277–282, 1993.

Appendix A

MATLAB Code

The *MATLAB* code for the PDR system implementation described in section 4.1 where, orientation is calculated using raw IMU measurements, and ZUPTs and ZARUs are applied, and the threshold-based method is used for stance phase detection is given below:

```
clear all;


%% Read data from file and initialize parameters
data = dlmread('long_corridor/yk3.log', ' ');
size_before_trimming = size(data, 1);
data = unique(data, 'rows'); % Unique returns sorted lines so we need to re-sort them.
data = sortrows(data, 2);
size_after_trimming = size(data, 1);
sensor_id = data(:,1);
sample_counter = data(:,2);
timestamp = data(:,3)/1000;
temperature = data(:,4);
```

```
acc_b = data(:,5:7)';
gyro_b = data(:,8:10)';
orientation = reshape(data(:,11:19),[],3,3);
g = 9.8126869202; % Norm of gravity.

%% Initializing the INS estimates

t=1;

% initializing attitude using Xsens's initial attitude solution
C=squeeze(orientation(t,:,:));

% gyro_b1 is the gyroscopic sensor data in the body frame with the gyro
% biases removed (from gyro_b)
gyro_b1 = nan(3, size_after_trimming);
gyro_b1(:,t) = gyro_b(:,t);



% acceleration

% acc_b1 is the acceleration in the body frame with the accelerometer
% biases removed (from acc_b)
acc_b1 = nan(3,size_after_trimming);
acc_b1(:,t) = acc_b(:,t);
acc_n = nan(3,size_after_trimming);
```

```
vel_n = nan(3, size_after_trimming);
% initializing velocity
vel_n(:,t)=[0 0 0]';


pos_n = nan(3, size_after_trimming);
% initializing position
pos_n(:,t)=[0 0 0]';


C_prev=C;
%% Initializing the KF parameters

% state vector components
attitude_error=[0 0 0]';
gyro_biases=[0 0 0]';
pos_error=[0 0 0]';
vel_error=[0 0 0]';
acc_biases=[0 0 0]';

% state vector
delta_x=[attitude_error' gyro_biases' pos_error' vel_error' acc_biases']';
```

```
% error covariance matrix
P =diag([10^-4 10^-4 10^-4 0 0 0 0 0 0 10^-4 10^-4 10^-4 0 0 0]);
% system noise covariance matrix
Q=diag([10^-4 10^-4 10^-4 0 0 0 0 0 0 10^-2 10^-2 10^-2 0 0 0]);

%measurement matrix
H=[zeros(3,3) eye(3,3) zeros(3,3) zeros(3) zeros(3,3);
    zeros(3,3) zeros(3,3) zeros(3,3) eye(3) zeros(3,3)];

% measurement noise covariance matrix
R = 0.001*eye(6);



%% INS estimates and Complementary Kalman Filter

for t=2:size_after_trimming



    dt=timestamp(t)-timestamp(t-1);

    % INS estimates



    %accounting for acc biases
    acc_b1(:,t) = acc_b(:,t)-acc_biases;
```

```
gyro_b1(:,t) = gyro_b(:,t)-gyro_biases;

%skew-symmetric matrix for angular rates
    ang_rate_matrix=[0   -gyro_b1(3,t)   gyro_b1(2,t);
                gyro_b1(3,t)  0   -gyro_b1(1,t);
                -gyro_b1(2,t)  gyro_b1(1,t)  0];

% attitude update
C=C_prev*(2*eye(3)+(ang_rate_matrix*dt))/(2*eye(3)-(ang_rate_matrix*dt));

% for use in the velocity integration step (next step), the acceleration must be
% resolved about the same axes as the velocity i.e. the navigation
% frame n and then the value of g is subtracted from the 'vertical'
% component of acceleration

acc_n(:,t) = C*acc_b1(:,t)-[0 0 g]';

% velocity estimation
vel_n(:,t) = vel_n(:,t-1) + (acc_n(:,t)+ acc_n(:,t-1))*dt/2;


% position estimation
pos_n(:,t) = pos_n(:,t-1) + (vel_n(:,t) + vel_n(:,t-1))*dt/2;
```

```
    % Complementary Kalman Filter

    % although a KF can be used to estimate the state variables directly,
    % it is common in inertial navigation systems to instead use a
    % Complementary KF which operates only on the errors in the state variables

    % skew-symmetric cross-product operator matrix formed from the n-frame
    % accelerometer output vector used in the state transition matrix(below)
%     if norm(acc_b(:,t)) * norm(gyro_b(:,t)) < 5

    S=[0  -acc_n(3,t)  acc_n(2,t);
       acc_n(3,t)  0  -acc_n(1,t);
       -acc_n(2,t) acc_n(1,t) 0];


    % state transition matrix
    F = [eye(3) dt*C  zeros(3,3)    zeros(3,3)      zeros(3,3);
         zeros(3,3)     eye(3)      zeros(3,3)    zeros(3,3)      zeros(3,3);
         zeros(3,3)     zeros(3,3)       eye(3)  dt*eye(3)  zeros(3,3);
         -dt*S  zeros(3,3)        zeros(3,3)    eye(3)    dt*C;
         zeros(3,3)     zeros(3,3)        zeros(3,3)    zeros(3,3)      eye(3)];

    % Filter prediction

%       delta_x = F*delta_x; %not required
```

```
      P = F*P*F' + Q;

    % measurement model
%     The EKF gets feedback from measurements, only when the
%     persons foot is detected to be stationary on the ground (totally
%     still, or in a stance phase during walk).

    % ZUPT
    % Stance phase detection
    c1=0; %Condition 1
    c2=0; %Condition 2
    c3=1; %Condition 3
    s=3;
    acc_mag = sqrt(acc_b(1,t)^2 + acc_b(2,t)^2 + acc_b(3,t)^2);
    gyro_mag = sqrt(gyro_b(1,t)^2 + gyro_b(2,t)^2 + gyro_b(3,t)^2);
    if acc_mag>=9 && acc_mag<=11
        c1=1;
    end
    if gyro_mag<1.5
        c2=1;
    end
    if(t>16 && t<size_after_trimming-16)
    acc_local_mean = acc_b(:,t-s:t+s);
    acc_local_m=mean(acc_local_mean,2);
```

```
 acc_local_variance=acc_b(:,t-s:t+s);
for q=1:(2*s+1)
acc_local_variance(1,q)=(acc_local_variance(1,q)-acc_local_m(1))^2;
acc_local_variance(2,q)=(acc_local_variance(2,q)-acc_local_m(2))^2;
acc_local_variance(3,q)=(acc_local_variance(3,q)-acc_local_m(3))^2;
end
acc_local_v=mean(acc_local_variance,2);
acc_local=sqrt(acc_local_v(1)^2 + acc_local_v(2)^2 + acc_local_v(3)^2);
if acc_local<15
    c3=1;
else
    c3=0;
end
end
 if c1==1 && c2==1 && c3==1

     % Measurement vector
     z1 = gyro_b1(:,t)-[0 0 0]';
     z2 = vel_n(:,t)-[0 0 0]';
     z=[z1 ;z2];

     %Kalman gain
     K = (P*(H)')/((H)*P*(H)' + R);
```

```
    % Filter update

    delta_x = K*(z);

    P = (eye(15) - K*(H)) * P * (eye(15) - K*(H))' + K*R*K';


%end of Kalman Filter

% INS updates

attitude_error=delta_x(1:3);
gyro_biases=delta_x(4:6);
pos_error=delta_x(7:9);
vel_error=delta_x(10:12);
acc_biases=delta_x(13:15);



% after each measurement update, the non-bias error terms of
% the filtered state vector, delta_x, are reset to zero after the INS
% uses them to refine the current attitude, velocity and position.
% This is because those errors are already compensated and
% incorporated into the INS estimations. The only terms that
% are maintained over time in the EKF filter are the gyro and
```

```
    % accelerometer biases.

    % Refinement of position and velocity based on Kalman error estimates
         vel_n(:,t)=vel_n(:,t)-vel_error;

         pos_n(:,t)=pos_n(:,t)-pos_error;

    % skew-symmetric matrix for small angles

    ang_matrix = -[0   -attitude_error(3,1)   attitude_error(2,1);
                   attitude_error(3,1)  0   -attitude_error(1,1);
                   -attitude_error(2,1)  attitude_error(1,1)  0];


    C = C*(2*eye(3)+(ang_matrix))/(2*eye(3)-(ang_matrix));

    %Resetting the state vector to zero
    delta_x = zeros(15,1);
    end

    C_prev=C;

end

%% Plot 2D traces
```

```
figure;
hold on;
plot(pos_n(1,:),pos_n(2,:),'-r');
axis equal;
grid;

%% Rotate the 2D traces
figure;
hold on;
angle=105;
rot_mat=[cosd(angle) -sind(angle);
         sind(angle) cosd(angle)];
pos_rot=zeros(2,size_after_trimming);
for i=1:size_after_trimming
    pos_rot(:,i)=rot_mat*[pos_n(1,i) pos_n(2,i)]';
end
     plot(pos_rot(1,:),pos_rot(2,:),'r');
axis equal;
grid;



%% Plot 2D velocity
figure;
hold on;
v = sqrt(vel_n(1,:).^2 + vel_n(2,:).^2);
```

```
plot((timestamp-timestamp(1)),v,'r');


%% Plot altitude


figure;
hold on;
plot((timestamp-timestamp(1)),pos_n(3,:));
```