ISSN 0280-5316
ISRN LUTFD2/TFRT--5912--SE

Improving the Inertial Navigation System of the CV90 Platform Using Sensor Fusion

Johan Ambrius
Jimmie Jönsson

Lund University
Department of Automatic Control
December 2012

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
January 2013

Document Number
ISRN LUTFD2/TFRT--5912--SE

Author(s)
Johan Ambrius
Jimmie Jönsson

Supervisor
Erik Nygård, BAE Systems, Karlskoga, Sweden
Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle
Improving the Inertial Navigation System of the CV90 Platform Using Sensor Fusion

Abstract
The aim of this thesis was to synthesize and evaluate an inertial navigation system (INS) for the Combat Vehicle 90 Platform. The INS that was created utilize sensor fusion in order to combine the different signals coming from the vehicle's multitude of sensors to estimate the vehicle's position and heading in some known global reference frame. The CV90 standard INS, the NAV90 system, had performed the task of navigation with an unpredictable behavior due to the fact that it relied on heading estimates from a magnetic compass that is strongly influenced by nearby metallic objects, e.g. other vehicles. It will be demonstrated in this thesis that with a two-axis gyroscope mounted on the weapon's rotational axis, the position and heading estimate from the INS can continue to provide reliable information even during long periods with without GPS signal reception.

Keywords
inertial navigation system, Kalman filter, sensor fusion, NAV90, CV90

Classification system and/ or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-106

Recipient's notes

Security classification

Improving the Inertial Navigation System of the CV90 platform using Sensor Fusion

A M.Sc. Thesis

Authors
Johan Ambrius
Jimmie Jönsson

Advisor
Erik Nygård

Examiner
Rolf Johansson

Department of Automatic Control
December 2012

LUND
UNIVERSITY

Abstract

The aim of this thesis was to synthesize and evaluate an inertial navigation system (INS) for the Combat Vehicle 90 Platform. The INS that was created utilize sensor fusion in order to combine the different signals coming from the vehicle's multitude of sensors to estimate the vehicle's position and heading in some known global reference frame. The CV90 standard INS, the NAV90 system, had performed the task of navigation with an unpredictable behavior due to the fact that it relied on heading estimates from a magnetic compass that is strongly influenced by nearby metallic objects, e.g. other vehicles. It will be demonstrated in this thesis that with a two-axis gyroscope mounted on the weapon's rotational axis, the position and heading estimate from the INS can continue to provide reliable information even during long periods with without GPS signal reception.

Keywords: inertial navigation system, Kalman filter, sensor fusion, NAV90, CV90

Table of Contents

List of Figures

List of Tables

Acknowledgments

The success of this project is the result of the combined knowledge and experience of many different people and I would like to extend my sincere thanks and gratitude to all of them.

My friend Jimmie Jönsson was a dependable colleague and roommate, and I am glad to have worked with him during this thesis project. Our stay in Karlskoga was a very pleasant experience, which was in large part due to Pär Eriksson, who never hesitated to extend a helping hand. The work environment at BAE Systems was excellent and the friendly atmosphere always made us feel welcome. Working there was very informative and educational thanks to the sage advice frequently offered to us by our advisor Erik Nygård and our colleague Boyko Iliev. Our supervisor Rolf Johansson made sure we stayed on the right path throughout the project and provided valuable input and suggestions.

Finally, I would like to thank my mother Carina, my father Jonny, and my sister Pernilla, for their love and support, without which I would not be where I am today.

Johan Ambrius
Lomma, 20 December 2012.

More than five years ago I arrived in Lund to pursue a master's degree in engineering. The path I had chosen was not without its trials, and I was tempted to give in on more than one occasion, but with the help of an incredible group of friends and family I made it. I owe a debt of gratitude to several people, not least my friend and coworker Johan Ambrius, who made it possible for me to complete my Masters of Engineering thesis.

I would like to thank our advisor Erik Nygård for all his help and valuable insight. More than once he casually suggested a solution that had eluded both Johan and me. I would also like to thank the rest of the nice people at BAE Systems in Karlskoga who made us feel right at home.

Finally, I would like to thank the Department of Automatic Control at Lund University and especially professor Rolf Johansson.

Jimmie Jönsson
Lund, 25 December 2012.

1 Introduction

The whole is greater than the sum of its parts.

Aristotle (384 BC - 322 BC)
Greek philosopher and polymath.

In an active combat zone it is extremely important for a vehicle to be able to determine its exact location since a reliable position estimate could mean the difference between life and death. As most combat vehicles around the world are equipped with a Global Positioning System (GPS), which has become the *de facto* standard in accurately pinpointing one's current position, this usually is not a problem. The concern is, however, what happens when the connection to the GPS satellite is lost.

Today, the Combat Vehicle 90 platform (CV90) is outfitted with an inertial navigation system (INS) that is highly dependent on its compass, which is very sensitive to magnetic disturbances such as other vehicles, scrap metal, containers and even the vehicle's own metallic components. This has been known to pose a problem when navigating without GPS support in an urban environment, such as in Afghanistan.

The goal of this thesis is to develop and propose a new kind of INS using sensor fusion that relies more on already available data from the two-axis gyroscope and angle sensors, and that is able to navigate for longer periods of time without constant GPS satellite contact.

2 The CV90 Platform

Figure 2.1: CV90 with 3D camouflage, front view. [13]

The CV90 platform is a family of light armored vehicles designed and manufactured in a joint venture between BAE Systems Hägglunds AB and Saab Bofors AB. The vehicles all share a common chassis design as well as component designs, differing mostly in weapon systems and special equipment, which contributes to a low life-cycle cost. The development began after the decisions of the Swedish Ministry of Defence in 1977 and 1982 to increase the mechanization within the Swedish Armed Forces [8].

Figure 2.2: CV90 with 3D camouflage, front-side view. [14]

The CV90 family is highly adaptable and can perform a multitude of different functions, and this is one of the reasons for the platform's success among many armies around the world. Denmark, Finland, The Netherlands, Norway and Switzerland have ordered a total of 576 vehicles, while the Swedish Armed Forces currently have 509 vehicles. Most implementations of the vehicle are basic light armored infantry vehicles designed to engage infantry fighting vehicles, other ground targets and helicopters. Other models assume the roles of e.g. fire-control; combat management; electronic warfare; towing and anti-air. The engine is a 550 HP Scania DSI14, giving a top speed of 70 km/h, and the latest generation of vehicles has a range of 600 km. Common armaments include a Bofors 40 mm gun, a coaxial 7.62 mm machine gun as well as six grenade launchers intended for smoke grenades [20].

3 Theoretical Background

Almost all of the sensor data generated by the combat vehicle need to be processed in some way. While global position and speed measurements may be handled directly, others, e.g. gyroscope readings, require additional consideration.

3.1 Cartesian Coordinate Systems

A system where numbers are used to uniquely determine the position of a geometric element on a manifold, such as Euclidean space, is called a coordinate system. Consequently, the numbers are called coordinates. The focus in this thesis will be the three-dimensional Cartesian coordinate system, where a position is specified by its signed distances to three mutually perpendicular planes spanned by the coordinate axes of the system [10]. An example of a right-handed, or positive oriented, Cartesian coordinate system can be seen in Fig. 3.1.

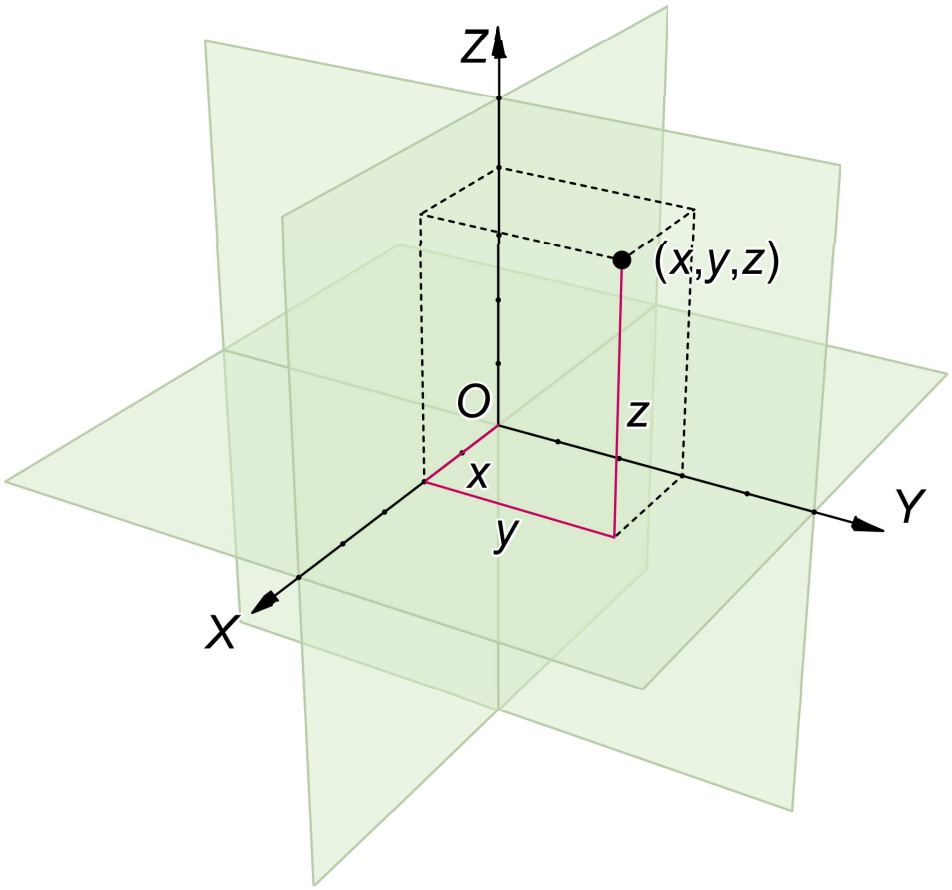

Figure 3.1: A three dimensional Cartesian coordinate system, with origin O and axis lines X, Y and Z, oriented as shown by the arrows. The dot marks on the axes are one length unit apart. Source: Jorge Stolfi [17].

In the coming sections the notation *frame of reference*, or *frame*, will refer to a coordinate system that describes points in space. For the purpose of this thesis the following frames are defined,

Ground Fixed North (Horizonted Frame), *Ground*
The positive x-axis points east, the positive y-axis points north and and the positive z-axis points up. The origin is placed in a point on the surface of the Earth.

Carrier
The x-axis is aligned along the vehicle body and is positive in the direction of movement, the y-axis is defined in a positive Cartesian coordinate system and the positive z-axis is perpendicular to the carrier body and points upward. The origin is placed in the mass center of the vehicle.

Turret/Upper Mount
The x-, y- and z-axes are parallel to the axes in the Weapon Frame when all rotations are zero. The origin is placed in the weapon's center of rotation.

Sensor/Weapon
The x-axis is aligned along the weapon and is positive in the fire direction, the positive y-axis points to its left and the positive z-axis points upward. The origin is placed in the center of rotation.

The point of interest is the position of the vehicle on the ground, which is why some kind of transformation between the different frames must be devised. Some measurements must also be transformed to the ground frame before they may be used as a comparison tool as the meaning of the measurements is lost otherwise.

3.2 Euclidean Coordinate Transformations

A mapping of points in the Euclidean plane to themselves which preserve distance is a called an Euclidean transformation. While there exist more types of these mappings, this thesis will focus on *translations* and *rotations*.

The translation of a point (x, y) is the equivalent of adding a fixed pair of numbers (a, b) to it, i.e. the new coordinates will be $(x', y') = (x + a, y + b)$, while the rotation by some angle θ counterclockwise around the origin is equivalent to the new coordinates $(x', y') = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$. A translation followed by a rotation is not always equivalent to a rotation followed by a translation, which is illustrated in Fig. 3.2 and Fig. 3.3.

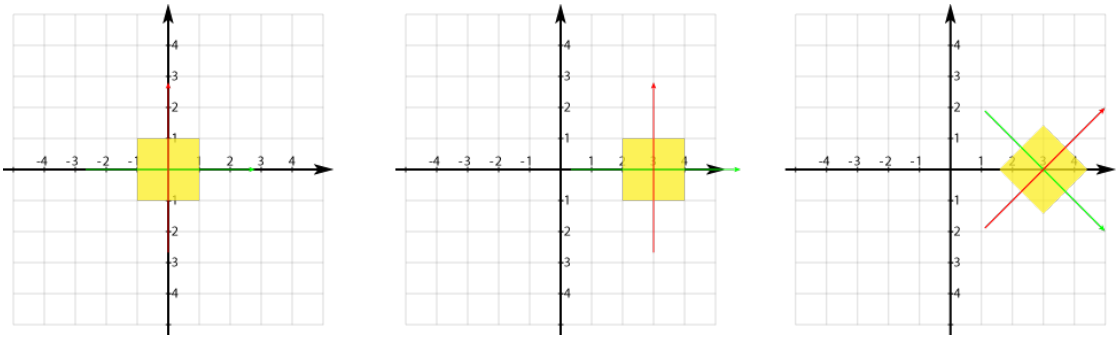

Figure 3.2: Example of translation and rotation of a system. The original position of the square is in the left plot, it is translated along the red arrow of the system in the middle plot and, finally, it is rotated clockwise $\pi/4$ rad in the right plot. Source: Per-Erik Bergman [4].

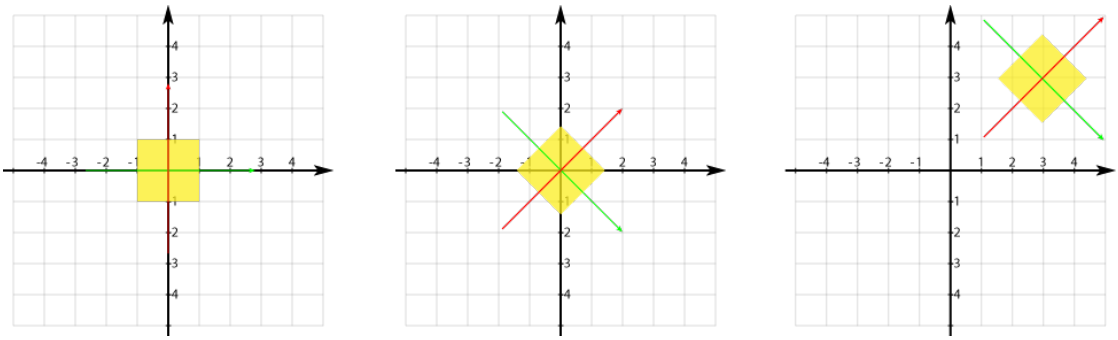

Figure 3.3: Example of rotation and translation of a system. The original position of the square is in the left plot, it is rotated clockwise $\pi/4$ rad in the middle plot and, finally, translated along the red arrow of the system. Source: Per-Erik Bergman [3].

Rotations in Euclidean space can also be expressed by matrices, thus providing a simple algebraic description of such rotations, [5]. By introducing the rotation matrix R

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{1}$$

and the column vector $\boldsymbol{p}$

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2}$$

the rotation by some angle θ counterclockwise around the origin can be described as the matrix multiplication $\boldsymbol{p}' = \mathbf{R}\boldsymbol{p}$. All Euclidean transformations can be represented using matrix multiplication if an extra dimension is added to $\mathbf{R}$, thus making the transformation linear. This is not favored, however, from a computational standpoint as it introduces many unnecessary operations.

To describe the orientation of a frame $\{B\}$ in relation to a known reference frame $\{A\}$ the notion of rotation matrices can be generalized to three dimensions. By starting from $\{A\}$ and performing successive rotations around the axes X, Y and Z respectively, the orientation of $\{B\}$ can be found. The derivation of the equivalent rotation matrix R_{XYZ} is straightforward and is given as

$$\begin{aligned}\mathbf{R}_{XYZ}(\gamma,\beta,\alpha) &= \mathbf{R}_Z(\alpha)\,\mathbf{R}_Y(\beta)\,\mathbf{R}_X(\gamma) \\ &= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix}, \end{aligned} \tag{3}$$

where $c_x = \cos x$, $s_x = \sin x$, γ is the rotation about X, β is the rotation about Y and α is the rotation about Z. The roll, pitch and yaw angles in Eq. (3) are γ, β and α, respectively, and are illustrated in Fig. 3.4.

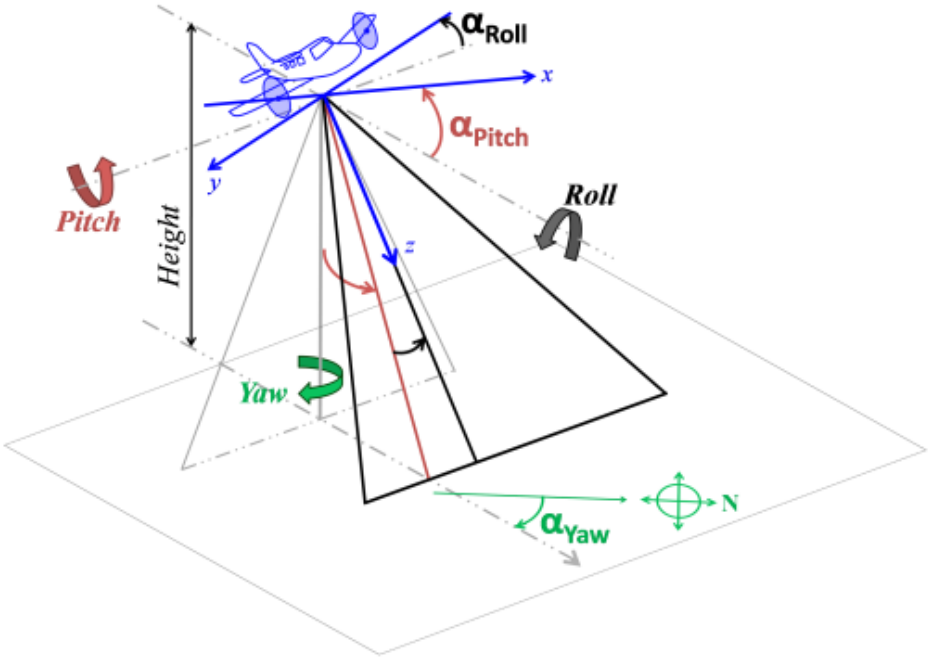

Figure 3.4: Roll, pitch, and yaw rotations and their corresponding angles. Source: Anja Schönhardt [16].

As each of these three rotations takes place about an axis in the fixed reference frame $\{A\}$ the convention for specifying the orientation is called X-Y-Z fixed angles. Furthermore, as rotation matrices are orthogonal matrices the relation $\mathbf{R}^T = \mathbf{R}^{-1}$ holds, a useful fact when the inverse transformation $\boldsymbol{p} = \mathbf{R}^{-1}\boldsymbol{p}'$ is calculated.

4 Available Sensors

CV90 is equipped with a number of different sensors, some were part of the original design while others have been added on later as force multipliers, e.g. the two-axis gyroscope for weapon stabilization and the GPS receiver for obtaining exact position information. The sensors are connected internally in the vehicle according to the diagram in Fig. 4.1, and are presented in more detail in the following subsections.

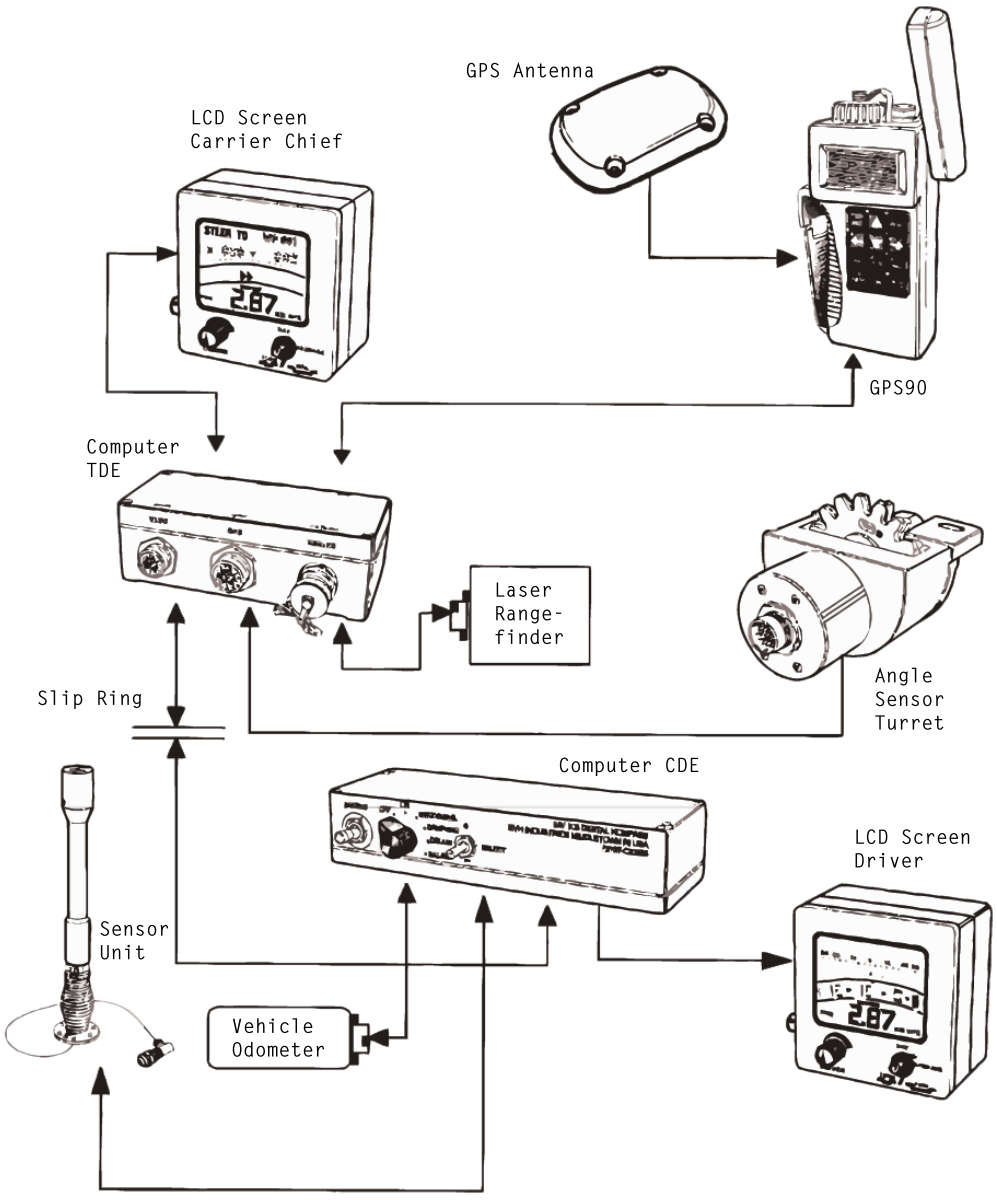

Figure 4.1: A graph showing the various sensors in the vehicle and how they are interconnected. Source: Erik Nygård [12].

4.1 Compass

The compass is located at the top-back of the carrier frame and is, as expected, very sensitive to changes in the magnetic field in its surroundings. During certain circumstances this has become a problem. Specifically, when the weapon is in the vicinity of the compass, i.e. pointing back across the carrier, it interferes with the compass' reading of the earth's weak magnetic field. Problems also arise when the vehicle is close to any large quantities of metal, e.g. another vehicle or a large metal container.

The first problem has been circumvented somewhat by the algorithms in the NAV90 system, which filters all the compass data before it does any calculations with it. These algorithms, however, can not remove any errors that occur when the vehicle is in the vicinity of other large structures of metal. This uncertain behavior can cause initial heading estimations to be incorrect, and since this is an important function to have for a new INS, this suggests that the compass is a sensor which should be improved, or replaced, in the future.

4.2 Angle Sensors Turret/Weapon

These sensors measure the angular position of the turret in relation to the carrier, and of the weapon in relation to the turret. Both give measurements in radians and the on-board computer in the vehicle also calculates the angular velocities of the respective angles.

The sensors themselves are mounted on the gears that manipulate the turret and weapon structures and are therefore mechanically linked with what they measure, which make them quite reliable.

4.3 Odometer

The odometer is placed in the gears in the drive train that is connected to the engine and consists of a Hall Effect sensor that emits a pulse when a gear tooth has passed it, and the on-board computer then counts the amount of cogs that has passed in total. The resolution of the odometer is 2.5 cm, i.e. the vehicle has to move at least 2.5 cm for the odometer to register a change in the distance traveled, which is a restriction imposed by the size of the cogs.

The odometer in CV90 not only measures distance but also calculates a speed, and so it also functions as a speedometer and at the beginning of this thesis it was uncertain if the odometer could handle neutral and reverse gear, i.e. give zero speed in neutral gear and a negative speed in reverse gear. However, it was found that the sensor could indeed handle these cases, see Sec. 9.4.3, which simplified the data extraction of the odometer signals since no flags had to be checked in order to deduce in which gear the vehicle was currently engaged.

4.4 Two-axis Gyroscope

The two-axis gyroscope is a Fiber Optic Gyroscope (FOG) mounted concentrically with the weapon's rotation axis and measures angular rotation by utilizing the *Sagnac effect*. The physical system consists of a light source, e.g. a diode laser, a detector to measure the interference, and a fiber optic coil which may have several loops, see Fig. 4.2.

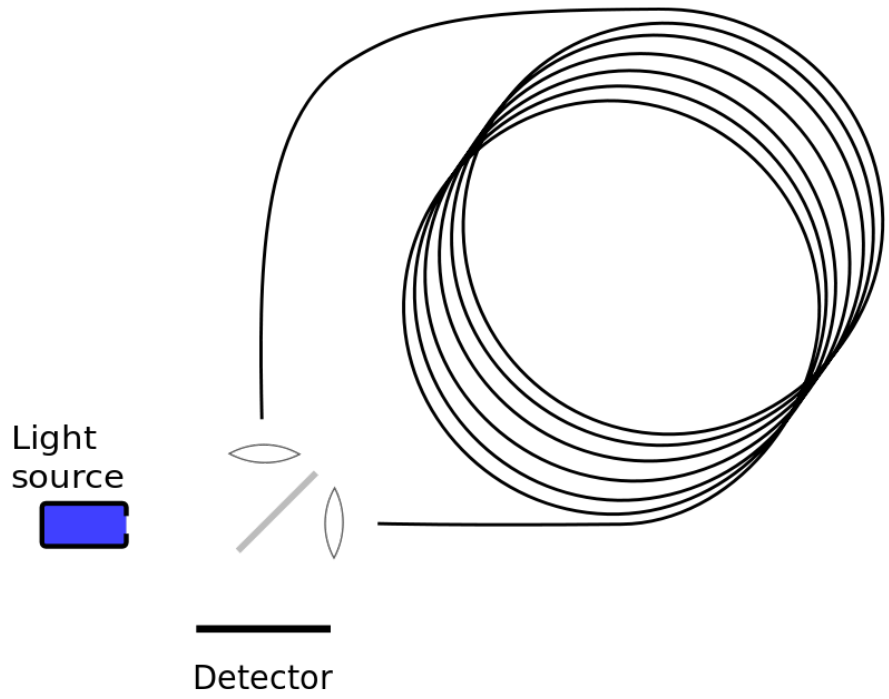

Figure 4.2: A diagram showing the general structure of a fiber optic gyroscope. Source: D. McFadden [9].

Two laser beams are injected into opposite ends of the fiber coil by beam splitting optics. The beams then exit the coil and if there has been any angular rotation in the FOG's reference frame, one beam will have experienced a slightly shorter path delay than the other beam, since they traveled in opposite directions. This phenomena is called the Sagnac effect and is the electromagnetic counterpart of the mechanics of rotation. A detector at the exit measures the interference of the beams and produces a value signifying how much any angular velocity has affected the system [19].

Since the strength of the Sagnac effect depends upon the effective area of the closed optical path, a higher quantity of loops provide better measurements. Due to this fact the fibre optic cables in FOG's are usually very long, sometimes as long as 5 km.

4.5 GPS receiver

The GPS receiver used during most of the testing was a standard GPS90 unit with a resolution of 1 meter in all three spatial dimensions. A u-blox GPS was used during the third testing day in Skövde on 2012-11-15, Sec. 9.4. The u-blox unit had better performance than the GPS90 system, both in spatial resolution as well as in the ability to find GPS satelittes.

A GPS receiver estimates its position by timing the signals broadcast by the network of GPS satellites, called the GPS constellation, that orbit the earth. The signals contain the time at which the message was transmitted as well as the satellite position at that time. With this information the receiver can calculate the transit time of each message and compute the distance to each satellite using the speed of light. These distances are then interpreted as a number of radii which in turn defines a collection of spheres, where the receiver is located at the intersection of these spheres.

The location is calculated by using the *navigation equations*, which can be done both analytically and numerically depending on the implementation. In standard operation, four or more satellites must be visible to get accurate results. This restriction can be relaxed to three satellites when one variable is known, e.g. the altitude.

There exists several global navigation systems, e.g. GPS (USA), GLONASS (Russia) and Galileo (EU). The tests done during this thesis used the network of 30 GPS satellites, orbiting at an altitude of approximately 20,200 km, to determine the vehicle's position.

4.6 NAV90

NAV90 has been the *de facto* INS for the CV90 platform for most of the platform's time in service and it makes use of all the previously named sensors, except the gyroscope and GPS, as can be seen in Fig. 4.1.

While this thesis was meant to find an alternative to this system, some information that the NAV90 provides was utilized as initial conditions when running simulations. The information consisted of the carrier heading, roll and pitch. The intricacies of the NAV90 system are not covered in this thesis.

5 A Mathematical Model of CV90

A MODEL THAT DESCRIBES THE CV90 is needed in order to simulate how the vehicle behaves. By using sensor data from the odometer, the two-axis gyroscope and the side and elevation angle sensors, the movement in the carrier frame can be described. The movement in the ground frame is given by the transformation

$$\boldsymbol{v}' = \mathbf{R}_{XYZ}(\gamma, \beta, \alpha)\boldsymbol{v}, \tag{4}$$

where the velocity vector $\boldsymbol{v}$ is

$$\boldsymbol{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T, \tag{5}$$

and where the transformation matrix $\mathbf{R}_{XYZ}$, see Eq. 3, is determined by how the carrier frame is rotated in relation to the ground frame. As the carrier only can travel along its x-axis, Eq. 5 becomes

$$\boldsymbol{v} = \begin{bmatrix} v_x & 0 & 0 \end{bmatrix}^T. \tag{6}$$

The ground velocities of the carrier can thus be expressed as

$$\boldsymbol{v}' = \mathbf{R}_{XYZ}(\gamma, \beta, \alpha)\boldsymbol{v} \tag{7}$$

$$\begin{aligned} &= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix} \begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} c_\alpha c_\beta \\ s_\alpha c_\beta \\ -s_\beta \end{bmatrix} v_x, \end{aligned} \tag{8}$$

where $c_x = \cos x$ and $s_x = \sin x$. By Eq. 8 the movement in the ground coordinates x, y, z are therefore

$$\dot{x} = v_x \cos\alpha \cos\beta \tag{9}$$

$$\dot{y} = v_x \sin\alpha \cos\beta \tag{10}$$

$$\dot{z} = -v_x \sin\beta. \tag{11}$$

For some purposes the ground frame might be defined with the positive z-axis pointing downward, which is why a second rotation must be applied to Eq. 4 as

$$\boldsymbol{v}' = \mathbf{R}_X(\pi)\mathbf{R}_{XYZ}(\gamma, \beta, \alpha)\boldsymbol{v}, \tag{12}$$

with the end result

$$\dot{x} = v_x \cos\alpha \cos\beta \tag{13}$$

$$\dot{y} = -v_x \sin\alpha \cos\beta \tag{14}$$

$$\dot{z} = v_x \sin\beta. \tag{15}$$

The angular velocity of the carrier around the z-axis in the ground frame is the difference between the gyroscope measurement around this axis and the angular velocity of the turret, computed from the side angle sensor. Similarly, the angular velocity around the y-axis is the difference between the gyroscope measurement around this axis and the angular velocity of the weapon, computed from the elevation angle sensor. In order to the keep the mathematical model from being overly complicated it is assumed that all coordinate transformations are completed, i.e. sensor data from the gyroscope is transformed to the carrier frame, and that the measurements from the angle sensors have been subtracted from the corresponding gyroscope axis. As a result, the angle sensors does not need to be included in the formulation, and the angular velocities of the carrier are simply given as

$$\dot{\alpha} = u_\alpha \tag{16}$$

$$\dot{\beta} = u_\beta, \tag{17}$$

where u_α is the gyroscope measurement around the z-axis and u_β is the gyroscope measurement around the y-axis, each in the ground frame, with the corresponding angles α and β. The carrier is only fitted with a two-axis gyroscope why it is not possible to estimate the angular velocity around the x-axis, i.e. the angle γ.

Measurements from the gyroscope are corrupted by various kinds of errors, see Sec. 6.2.1, and so two extra equations for each gyroscope axes should be included in the model, corresponding to the bias repeatability error and the bias stability error, respectively. Using Eq. 11 and Eq. 17 a mathematical model of CV90, where the gyroscope errors affect the angular velocities of the carrier, can be summarized as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{cases} \dot{x}_1 = v_x \cos x_4 \cos x_5 \\ \dot{x}_2 = v_x \sin x_4 \cos x_5 \\ \dot{x}_3 = -v_x \sin x_5 \\ \dot{x}_4 = u_\alpha - x_6 - x_7 \\ \dot{x}_5 = u_\beta - x_8 - x_9 \\ \dot{x}_6 = -\dfrac{1}{\tau} x_6 \\ \dot{x}_7 = 0 \\ \dot{x}_8 = -\dfrac{1}{\tau} x_8 \\ \dot{x}_9 = 0. \end{cases} \tag{18}$$

A natural choice of states is to let the Cartesian coordinates, the angles of rotation and the gyroscope errors be separate states. From the model, the position of CV90 is given by x_1, x_2 and x_3 while the orientation is given by x_4 and x_5. Due to the limitations of the gyroscope, nothing can be said about the roll angle of the vehicle.

5.1 Linearization

In order to use a linear Kalman filter with the derived state-space model, the nonlinear functions $f(\boldsymbol{x}, \boldsymbol{u})$ must be linearized by a Taylor series expansions around a stationary operating point $(\boldsymbol{x}_0, \boldsymbol{u}_0)$, [6]. As not all points of operation are stable, the system is linearized around the current estimate of the carrier angles with a zero velocity, i.e. $v_x = 0$. The Taylor series expansion is

$$f(\boldsymbol{x}, \boldsymbol{u}) \approx f(\boldsymbol{x}_0, \boldsymbol{u}_0) + \frac{\partial}{\partial \boldsymbol{x}} f(\boldsymbol{x}_0, \boldsymbol{u}_0)(\boldsymbol{x} - \boldsymbol{x_0}) + \frac{\partial}{\partial \boldsymbol{u}} f(\boldsymbol{x}_0, \boldsymbol{u}_0)(\boldsymbol{u} - \boldsymbol{u}_0), \tag{19}$$

and by introducing the new variables

$$\begin{aligned} \Delta \boldsymbol{x} &= \boldsymbol{x} - \boldsymbol{x}_0 \\ \Delta \boldsymbol{u} &= \boldsymbol{u} - \boldsymbol{u}_0, \end{aligned}$$

the linearized system can be written as

$$\Delta \dot{\boldsymbol{x}} = \mathbf{A} \Delta \boldsymbol{x} + \mathbf{B} \Delta \boldsymbol{u}. \tag{20}$$

The system matrix $\mathbf{A}$ is

$$\mathbf{A} = \frac{\partial f}{\partial \boldsymbol{x}} = \left[\begin{array}{c|c} \mathbf{A}_1 & \mathbf{A}_2 \\ \hline \mathbf{A}_3 & \mathbf{A}_4 \end{array}\right], \tag{21}$$

with the submatrices

$$\mathbf{A}_1 = \mathbf{0}_{3,3}, \qquad \mathbf{A}_2 = \begin{bmatrix} -v_x \sin x_4 \cos x_5 & -v_x \cos x_4 \sin x_5 & \\ v_x \cos x_4 \cos x_5 & -v_x \sin x_4 \sin x_5 & \mathbf{0}_{3,4} \\ 0 & -v_x \cos x_5 & \end{bmatrix}$$

$$\mathbf{A}_3 = \mathbf{0}_{6,5}, \qquad \mathbf{A}_4 = \begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ -\frac{1}{\tau} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\tau} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

while the input matrix $\mathbf{B}$ is

$$\mathbf{B} = \frac{\partial f}{\partial \boldsymbol{u}} = \left[\begin{array}{c} \mathbf{B}_1 \\ \hline \mathbf{B}_2 \end{array}\right], \tag{22}$$

with the submatrices

$$\mathbf{B}_1 = \begin{bmatrix} 0 & 0 & \cos x_4 \cos x_5 \\ 0 & 0 & \sin x_4 \cos x_5 \\ 0 & 0 & -\sin x_5 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{B}_2 = \mathbf{0}_{4,3}.$$

As only the position and the heading of the vehicle is measurable the matrix $\mathbf{C}$ becomes

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{23}$$

5.2 Discretizing the Linearized State Space Equations

Assuming that the system input $\boldsymbol{u}$ is piece-wise constant [7],

$$\boldsymbol{u}(t) = \boldsymbol{u}_{kh},\ t \in [kh, (k+1)h) \tag{24}$$

where h is the sample time, a linear system, such as the one described in Sec. 5.1, can be discretized into

$$\boldsymbol{x}_{kh+h} = \boldsymbol{\Phi}\boldsymbol{x}_{kh} + \boldsymbol{\Gamma}\boldsymbol{u}_{kh} + \boldsymbol{w}_{kh} \tag{25}$$

$$\mathbf{y}_{kh} = \mathbf{C}_d\boldsymbol{x}_{kh} + \boldsymbol{v}_{kh}, \tag{26}$$

where the covariances are $\boldsymbol{w}_k \sim \mathcal{N}(0, \mathbf{Q}_d)$ and $\boldsymbol{v}_k \sim \mathcal{N}(0, \mathbf{R}_d)$. The system matrices are discretized as [21]

$$\boldsymbol{\Phi} = e^{\mathbf{A}h} \tag{27}$$

$$\boldsymbol{\Gamma} = \int_{\tau=0}^{h} e^{\mathbf{A}\tau} d\tau \mathbf{B} \tag{28}$$

$$\mathbf{C}_d = \mathbf{C}. \tag{29}$$

The noises in the continuous-time model are modeled from the noises that appear in a discrete electro-mechanical system, and there is no need to discretize them again. This gives the simple relationship

$$\mathbf{Q}_d = \mathbf{Q} \tag{30}$$

$$\mathbf{R}_d = \mathbf{R}. \tag{31}$$

The initial state and the noise vectors at each time step are all assumed to be mutually independent.

6 Simulink Models

There are several components that have to be modeled in order to be able to introduce relevant errors and noise into the complete system,

Signal Generator
: Generates the accelerations and angular accelerations that are needed to drive the system.

Gyroscope
: Measures angular velocities around three orthogonal axes.

Odometer
: The odometer measures the distance traveled but it can also function as a speedometer.

GPS Receiver
: Acquires the ”true” position, i.e. true in the sense that it is the best position estimate available.

Angle Sensors
: Gives the horizontal position of the turret and the vertical position of the weapon.

The angle sensors have Simulink models but since they only add noise and quantizise the signal, they are trivial in complexity compared to the listed components and so they will not be covered in detail.

6.1 Signal Generator

To generate signals representing the real system in an efficient manner, a signal generator was implemented in the form of a Simulink model with a high sampling rate of 1 kHz in order to provide enough resolution. The model takes the three angular accelerations of the carrier, the angular accelerations of the turret and of the weapon, as well as the carrier's acceleration vector. This gives a total of 8 input signals that have to be defined before a simulation can be run. The output of the model are the input signals as well as their integrated and twice integrated signals, i.e. their velocities and distances, respectively, for a total of 24 possible output signals.

The frames in Sec. 3.1 are defined in order to establish how the generated signals act in their relevant coordinate system. The angular velocities are generated in different frames depending on their properties, e.g. the readings from the gyroscope are dependent on both the side angle of the turret and the elevation angle of the weapon which is why the signal is generated in the ground frame and then transformed to the real sensor frame.

Others, e.g. the side angle of the turret, directly correspond to the input in the signal generator and is thus in the correct coordinate system from the start. The carrier velocities can be used as directly since they already are in the correct reference frame.

6.2 Gyroscope Model

The gyroscope takes true values, i.e. untainted signals, of the angular velocities and filters them to produce estimated angular velocities. In the model the true signal is first passed through a low-pass filter, then passed through a sub block where the errors are added, and, finally, quantizied. An overview of the SIMULINK model of the gyroscope can be seen in Fig. 6.1.

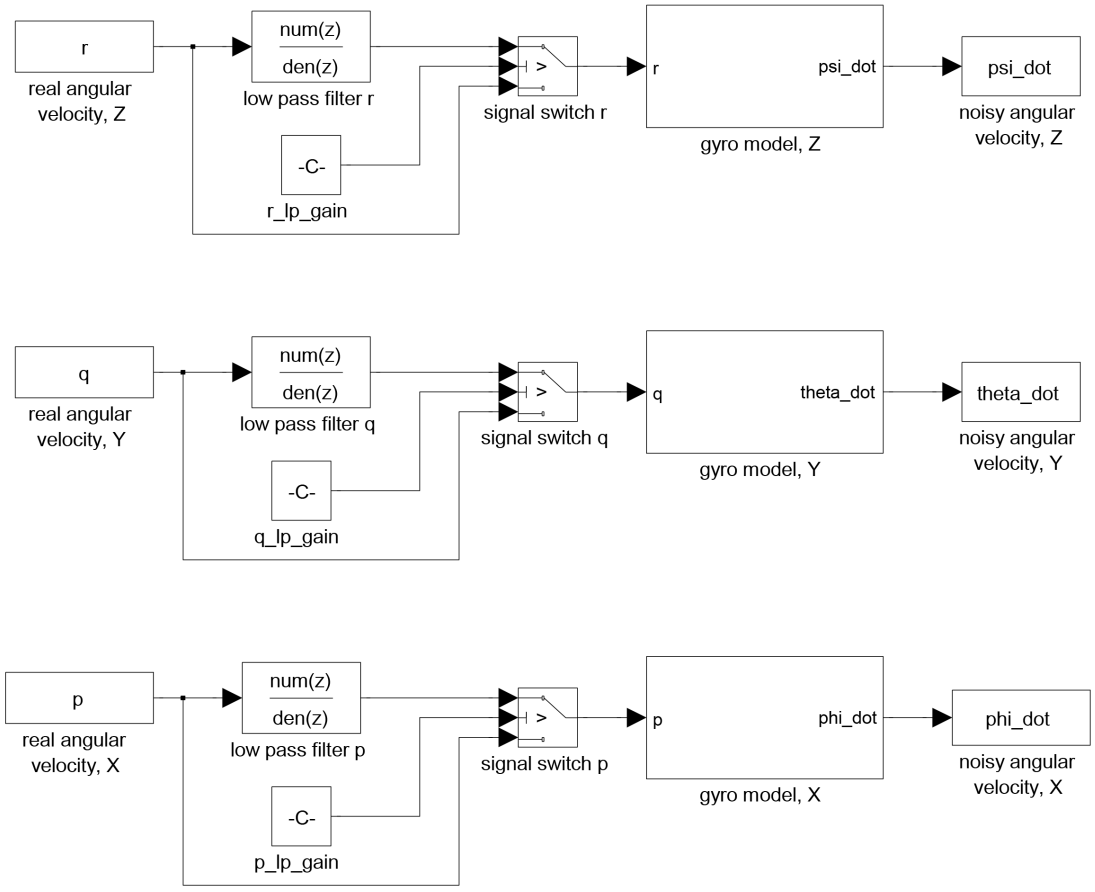

Figure 6.1: SIMULINK model of the three gyroscope channels. A variable in the switch for each channel controls whether the low-pass filtered signal or the unfiltered signal enters the main gyroscope block. This functionality was mainly used for model verification purposes.

6.2.1 Gyroscope Errors

The inputs will be low-pass filtered through a filter which has characteristics that are determined in the gyroscope's specification [15], and they will also be contaminated by noise of various types. Although many different noises and errors can be modeled, only the most significant error sources will be covered in this thesis, specifically: low-pass filtering; additive noise in the form of bias stability, bias repeatability, and random walk; and finally, quantization errors.

Other commonly specified errors that exist within gyroscopes, but which has not been taken into consideration here, are: scale factor errors; linearity errors; misalignment of axes; as well as linear and nonlinear sensitivity drift.

The three additive noises can be seen in Fig. 6.2. The quantization acts just before the signal is sent out of the sensor, see Fig. 6.3.

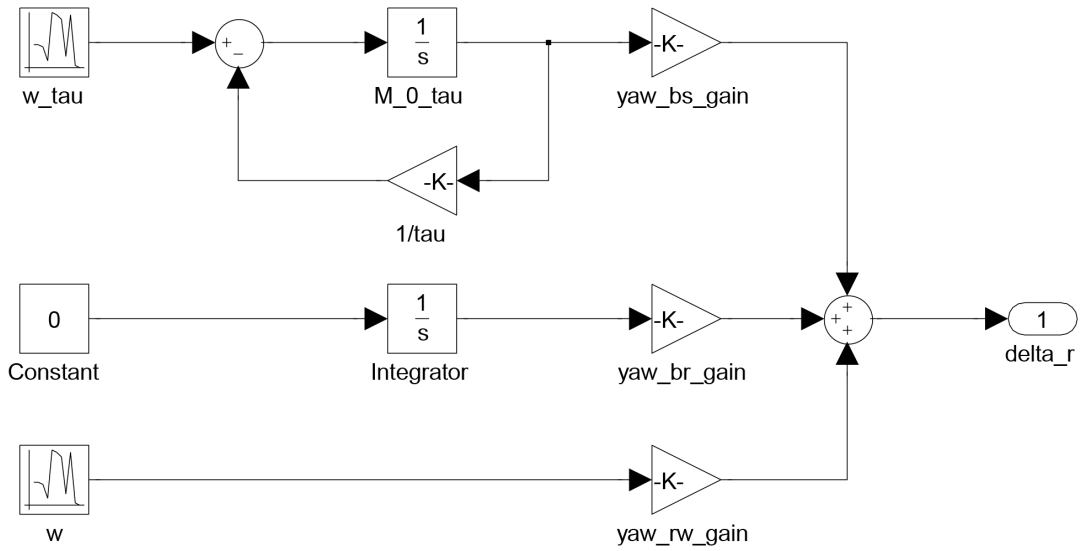

Figure 6.2: The additive error block for the yaw channel of the gyroscope. A similar block is present in the roll and pitch channels as well. The upper pipeline creates the bias stability error, the middle pipeline creates the bias repeatability error, and the lower pipeline creates the random walk error. All of the errors are summed and outside the block they are added to the true angular velocity signal.

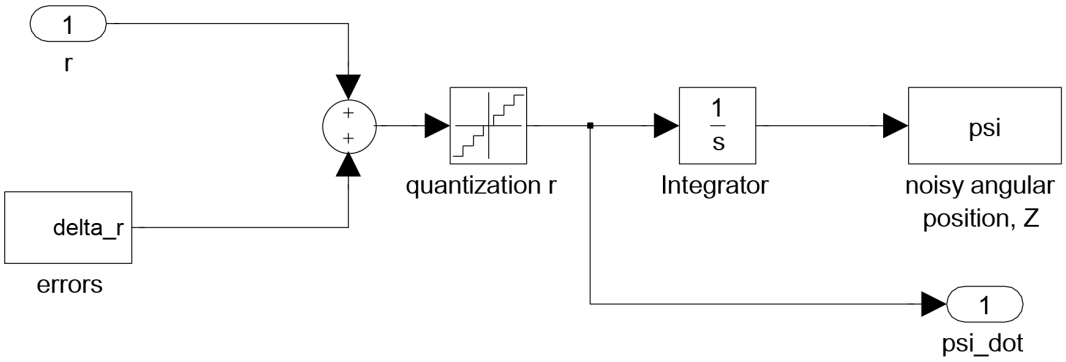

Figure 6.3: An inside view of the gyroscope model block for the yaw channel as seen in Fig. 6.1. The original angular velocity `r` is summed with the errors `delta_r` coming from the errors seen in Fig. 6.2. This noisy signal is then quantized and divided up into one velocity signal and one integrated position signal.

Low-pass Filter

According to [15], the low-pass filter will have a minimum bandwidth of 300 Hz when the phase lag is $-90°$, i.e. the gain at 300 Hz should be -6 dB. This is equivalent to a continuous second order system of the form

$$G(s) = \frac{3.553 \cdot 10^6}{s^2 + 3770s + 3.553 \cdot 10^6} \tag{32}$$

and will give the bode diagram shown in Fig. 6.4.

In order for this filter to be implemented in the SIMULINK model it has to be transformed to discrete form. Applying standard continuous to discrete transformations, with sample rate 1 kHz, yields the following discrete filter

$$H(z) = \frac{0.562z + 0.1574}{z^2 - 0.3037z + 0.02305} \tag{33}$$

which in the model is defined in the `low-pass filter` blocks seen in Fig. 6.1. The low-pass filter makes it impossible to accurately detect very fast changes in the system. It is, however, a highly improbable occurrence that the gyroscope should move at the angular speeds needed in order for this to become a real problem.

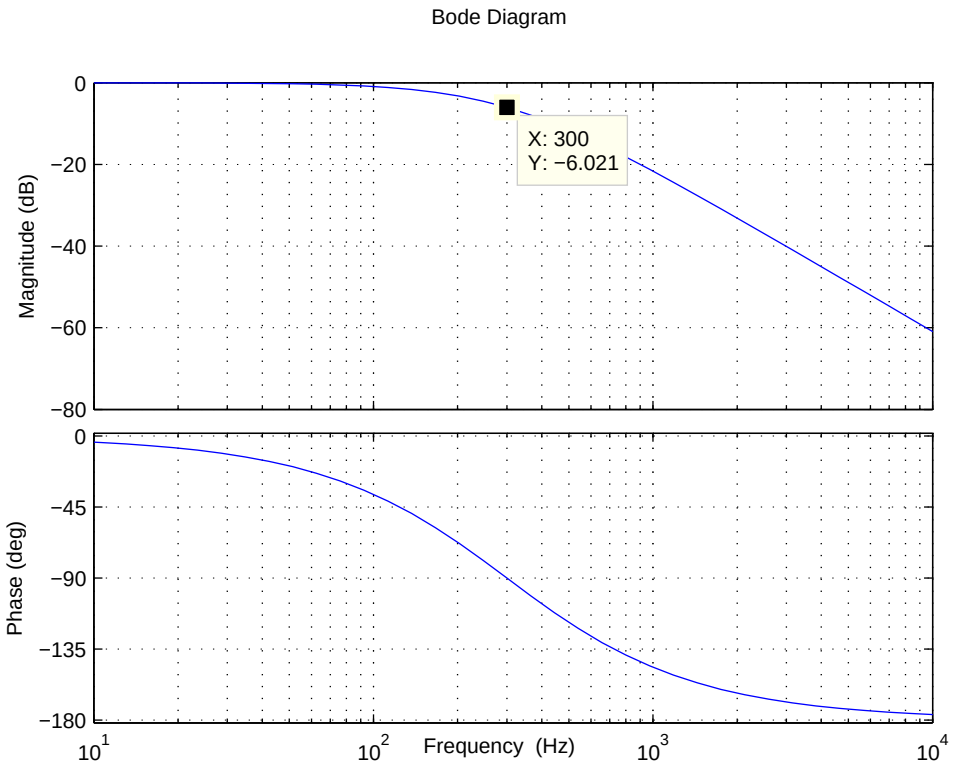

Figure 6.4: Bode plot of the continuous version of the low-pass filter that affects the gyroscope input signals.

Bias Stability Error

Bias stability is sometimes called *correlated* noise and is a measurement of how the gyroscope bias drifts over time and is in a sense one of the best ways to evaluate the performance of a gyroscope. It is modeled in the top signal pipeline in Fig. 6.2.

This error is modeled as an integrator driven by a combination of white noise and feedback of its own output divided by a correlation time τ. The driving noise has a variance depending on τ, [2],

$$\sigma_{w_\tau}^2 = 2 \cdot \frac{\sigma_{gbs}^2}{\tau}, \tag{34}$$

where σ_{gbs}^2 is the variance of the gyroscope's bias stability, so it is possible to vary its variance by changing τ. The initial condition of the integral is

$$M_{0\_\tau} = \sigma_{gbs}^2 \tag{35}$$

where, [15],

$$\sigma_{gbs} = \max 20\,^\circ/\mathrm{h} = \max 0.0056\,^\circ/\mathrm{s}. \tag{36}$$

Since τ was not given in the specifications, an educated guess was made and it was set to $\tau = 600$, which yields an initial condition of

$$M_{0\_\tau} = 9.4018 \cdot 10^{-9}\,\mathrm{rad}^2/\mathrm{s}^2. \tag{37}$$

The bias stability error gives 1 extra state for each gyroscop axis.

Bias Repeatability Error

Bias repeatability is sometimes called *random bias* and is included in the model in the middle signal pipeline in Fig. 6.2. This error creates a constant velocity bias each time the gyroscope is turned on and in this case it is specified, [2], as

$$M_{0\_Bias} = \sigma_{gbr}^2 \tag{38}$$

where σ_{gbr} is the gyroscope's bias repeatability which is defined, [15], as being

$$\sigma_{gbr} = \max 180\,^\circ/\mathrm{h} = \max 0.05\,^\circ/\mathrm{s} \tag{39}$$

This gives the integral an initial condition of

$$M_{0\_Bias} = 7.6154 \cdot 10^{-7}\,\mathrm{rad}^2/\mathrm{s}^2. \tag{40}$$

The bias repeatability error gives 1 extra state for each gyroscop axis.

Angular Random Walk

The angular random walk additive noise corresponds to integration of white noise and is represented in the model by the lower signal pipeline in Fig. 6.2.

The white noise error will affect the integrated signal by adding a bias to the variance of the output signal. This bias will be proportional to the square root of the integration time, which is the reason that the random walk specification is given as $^\circ/\sqrt{\mathrm{h}}$. In this case the value is $\sigma_{random\_walk} = \max 0.5\,^\circ/\sqrt{\mathrm{h}}$, [15]. This means that after 1 hour the added bias will be $0.5\,^\circ$ and after e.g. 100 hours it will be $0.5 \cdot \sqrt{100}\,^\circ = 5\,^\circ$.

The random walk error will not contribute to any new states in the state space model since it consists of pure white noise.

Quantization

Due to the finite resolution of the gyroscope, the input signals will become quantized to some extent. In this thesis it has been deemed appropriate to have a quantization level of 0.1°.

6.2.2 Testing the gyroscope Model

The error sources were set according to the specifications described earlier and two situations were tested. In the first the input signal represented one complete rotation around every axis, with constant angular velocity and a total rotation time of 10 s. The second test was the same as the first test except the time had been increased to 60 s. The results can be seen in Fig. 6.5.

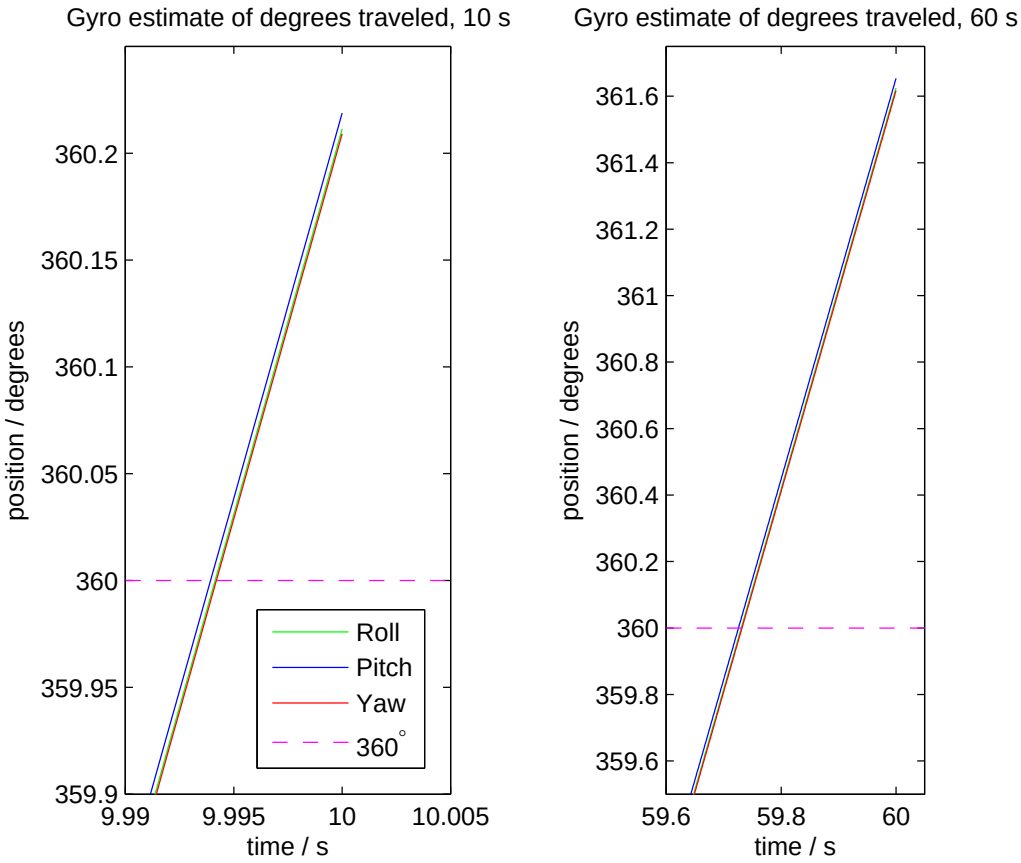

Figure 6.5: Two test cases for the gyroscope. The left plot shows the estimated end position when all three axes has performed one rotation during a time period of 10 s. The right plot shows the same test but with a time period of 60 s.

The filtered angles are not noticeably erratic during the simulation for either of the test cases, but the difference between the final values and the real end value of 360° differs between the two tests. These results reflect the performance of the gyroscope component and shows that while noise is low, there is still some drift present which increases with the integration time.

6.3 Odometer Model

The speed that is obtained from the odometer is used in conjunction with the velocity vector, given by the gyroscope, to calculate the carrier's velocity vector. The odometer SIMULINK model can be seen in Fig. 6.6.

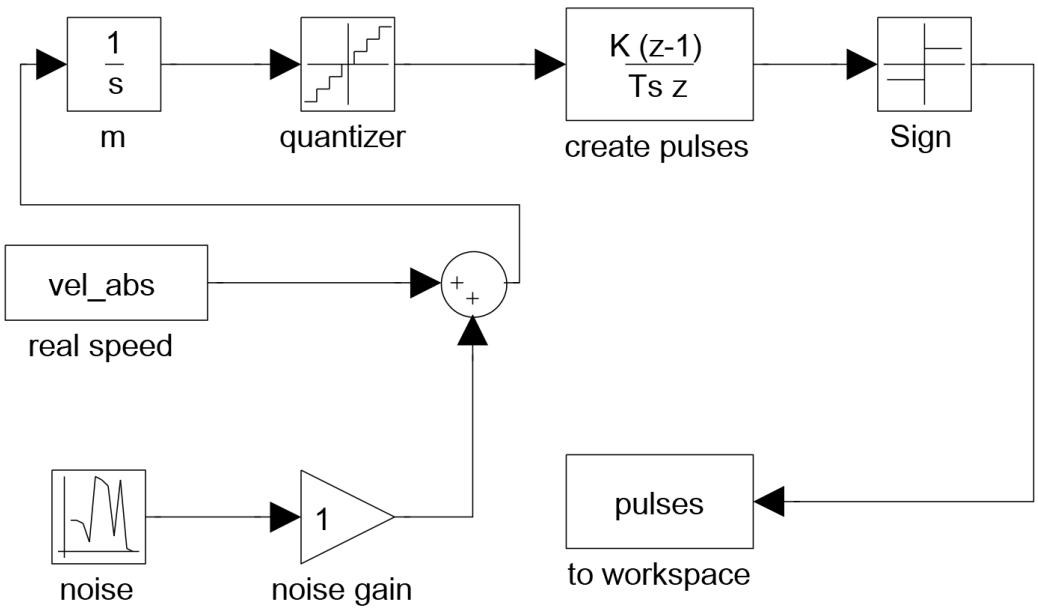

Figure 6.6: The odometer model takes the real speed of the carrier and adds noise. The signal is then integrated and quantized to get an approximation of the length traveled, which gives a stair-like curve. This curve is then differentiated and signed to create pulses which are sent back to the workspace and counted to produce a speed estimate.

6.3.1 Odometer Errors

Since the odometer had to be able to estimate speeds up to 70 km/h, the sampling rate of the odometer had to be relatively high. The number of pulses the odometer would have to detect if the vehicle was traveling at 70 km/h is

$$\frac{70}{3.6} \cdot \frac{1}{0.025} \approx 778. \tag{41}$$

Due to this the sampling rate of the odometer was set to 1 kHz. The model takes a defined true speed and adds normally distributed Gaussian noise, with an option to add offset as well. The signal is then quantized with resolution 2.5 cm and differentiated in order to get a pulse train. The pulse train is sent out to the workspace where the pulses are summed over a preset sample time, which in this case was set to 1 s, and thus an average velocity over that sample time is obtained.

Another error that could be present in the odometer is the inability to detect slip between the bands of the vehicle and the terrain, which could lead the INS to believe that it has traveled further than it actually has. This error was not modeled due to the fact that slip is a relatively complicated phenomenon to model and detect, and its impact on the precision of the simulations was deemed to be negligible compared to the error that would be introduced by quantization, offset and noise.

6.3.2 Testing the Odometer Model

A test track was created for the odometer in the form of an acceleration profile representing acceleration up to 70 km/h and then braking to 0 km/h, in a repeating pattern. The result of this test can be seen in Fig. 6.7.

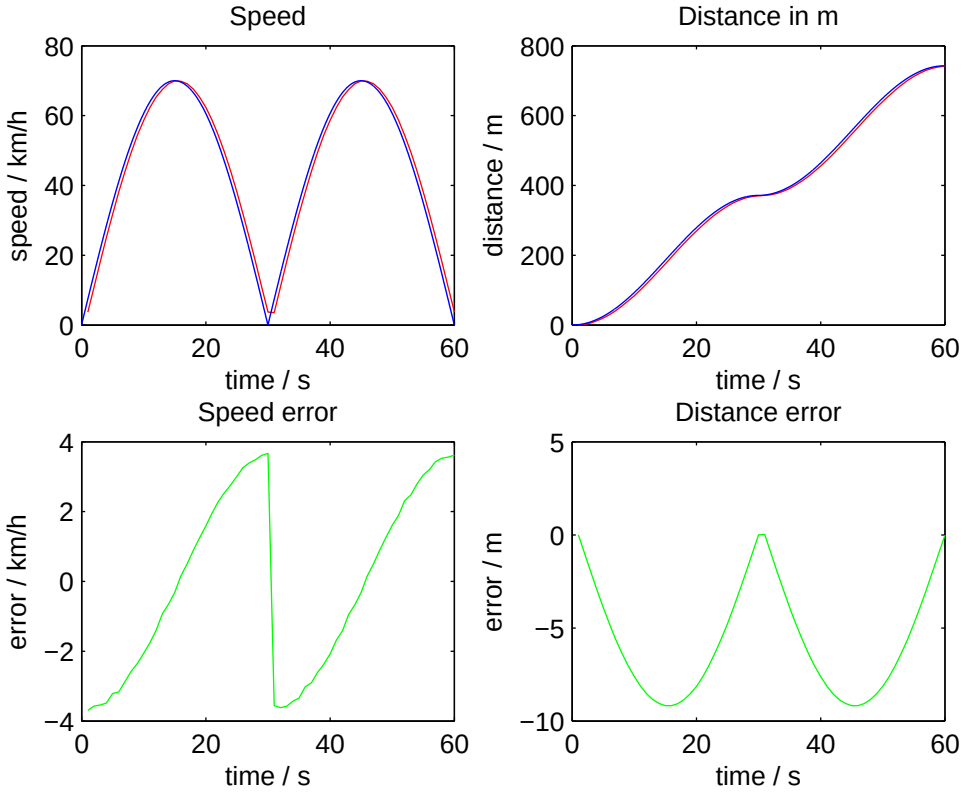

Figure 6.7: A test of the odometer model with a sampling rate of 1 Hz. The vehicle accelerated to 70 km/h and then decelerated to 0 km/h, and then repeated the procedure one more time. **Plot (1,1)** shows the true speed, in blue, and the estimated speed, in red. The 1 s delay is clearly visible and, depending on how the true speed fluctuates, it can cause both an overestimation and underestimation of the true speed. This difference is only observable when the carrier is accelerating or decelerating and does not appear at constant speeds. **Plot (2,1)** shows the difference between the true and estimated speeds. This error is either negative or positive depending on if the vehicle is accelerating or decelerating, respectively. **Plot (1,2)** displays the true distance, in blue, and the estimated distance, in red, of the carrier. The estimated value is slightly behind during acceleration and deceleration, however, it catches up during constant speeds. **Plot (2,2)** shows the difference between the true distance and the estimated distance. This error is always negative regardless of whether the carrier is accelerating or decelerating, but goes to zero when the carrier has constant speed.

6.4 GPS Model

The GPS model receives coordinates, with optional offset, with a set sample rate which describes the real world system, in this case set to 1 kHz. This information has normally distributed Gaussian noise added to it and is then quantized with a customizable resolution. The signal is then sampled at the GPS sample rate, which is set to 10 Hz, to represent how often the GPS takes information from the satellites.

This signal is then sampled again with the GPS refresh rate, which represents how often the GPS sensor sends information regarding position to the INS. The GPS refresh rate is usually set to 1 Hz, however, this is not usually a constant value due to the different priorities of the processes in the real time system in which the INS is implemented. This causes the 1 s sample time to fluctuate between certain values, in this case it has been set to be in the interval $t_{rr} \in [0.95, 1.05]$ s.

6.4.1 GPS Errors

Many factors have an impact on the precision of the GPS, such as the amount of available satellites it can receive information from, which can change drastically with weather and latitude, and the preset precision of information from the satellites, which is decided by those who control and maintain the satellites. Another error which is common in urban environments is so called *multipath effects*, which occur when GPS signals bounce of off nearby structures, causing the signal to be somewhat delayed and thereby introduce inaccuracies in the position estimate.

The GPS Simulink model circumvents the modeling of these problem by, as previously mentioned, simply taking the offset signals and not taking into consideration how they got the offsets.

6.4.2 Testing the GPS Model

The GPS model was tested and verified using different values for the offset, noise and resolution. All of the error values were based on information provided by BAE Systems. The model handled all the cases well and produced the expected output.

An interesting fact that was learned during the hunt for realistic values to use for testing was that in real GPS systems the z coordinate has a larger offset and error variance than the x and y coordinates. The reason for this is that all GPS satellites are not visible at all points on the globe, since the earth is very good at blocking their signals. The geometric pattern of the satellites that remain visible from one's current location makes it relatively easy to find the x and y coordinates, while the z coordinate will be harder to estimate.

7 Data Acquisition from the Real Vehicle

Real sensor data generated in real world conditions needed to be acquired in order to simulate the movement of CV90. The details of this enterprise are described in the following subsections.

7.1 Relevant Data

The signals needed to perform and evaluate the simulations of the system in Matlab are presented in Table 7.1.

Component	**Signal**	**Unit**
GPS	x coordinate	m
	y coordinate	m
	Reception status	bool
Gyroscope	Side Angle Velocity	rad/s
	Elevation Angle Velocity	rad/s
Angle Sensors	Side Angle Position	rad
	Side Angle Velocity	rad/s
	Elevation Angle Position	rad
	Elevation Angle Velocity	rad/s
Odometer	Velocity	km/h
NAV90	Time	s
	Pitch	rad
	Roll	rad
	Heading	rad
	UTC Time	s
	Laser Fired Status (PFP)	bool

table 7.1: table describing the signals that were extracted from the vehicle, as well as their units. The table also shows from which component the specified signal originated.

7.2 Data Extraction

The data were extracted from the vehicle using an Ethernet cable connected to a CAN bus port in the turret. The other end of the cable was connected to a laptop computer and the data were extracted using software developed by BAE Systems Bofors AB in Karlskoga.

The data were given in `.log` format where columns of data represented the different signals shown in the previous section. These files had to have their columns sorted in MATLAB since the signals were saved in a random order in the files. Another problem was the fact that it was only possible to log the internal data bus for a maximum of ca 10 minutes before a new log had to be started. This was due to limitations in the software, and it introduced gaps in some of the longer tests which made it necessary to adjust the heading manually after each log switch in order for the INS to follow the track properly.

8 The Kalman Filter

The main problem in this thesis consists of determining the precise current location of a CV90 using dead reckoning, i.e. using only old data to compute a new estimate. As the vehicle is expected to follow the laws of physics its position can be estimated by using the integrated velocity, obtained from the odometer, and the heading, derived from the gyroscope. Typically, dead reckoning will provide a very smooth estimate of the position, but it will drift over time as small errors from the integrations accumulate.

The vehicle is equipped with a GPS unit that also provides an estimate of the position within a few meters. The GPS estimate is likely to be noisy, i.e. readings will "jump around" rapidly, but will, however, always remain within a few meters of the real position. The proposed INS takes this randomneess into account and uses the GPS position to improve the dead reckoning position estimate.

8.1 Implementation

The Kalman filter was implemented in its recursive form. The main advantage of the recursive Kalman filter is its ability to use a series of measurements observed over time and to choose a statistically optimal weighting for these measurements. This usually produces estimates of unknown system variables that tend to be more precise than those that would be based on a single measurement alone. While the internal state often have more degrees of freedom than is observable, the Kalman filter can estimate the entire internal state by combining a series of measurements. The basic is idea is illustrated in Fig. 8.1.

The Kalman filter model assumes that the true state at time $k+1$ is evolved from the state at k according to the discretized system model in Eq. 25 and Eq. 26, where $\mathbf{\Phi}$ is the state transition matrix, $\mathbf{\Gamma}$ is the input matrix, and where $\boldsymbol{w}_{kh}$ and $\boldsymbol{v}_{kh}$ are the process and measurement noise, respectively, are assumed to be drawn from a zero mean multivariate normal distribution.

The main phases of the Kalman filter are "Predict" and "Update". The predict phase produces an estimate of the state at the current time step by utilizing information from the previous time step. This is why this estimate is called the *a priori* state estimate because it does not include observation information from the current time step. The update phase refines the state estimate by combining the *a priori* estimate with information of the system from the current time step. This new estimate is called the *a posteriori* state estimate. The details of the algorithm are described below with the assumption that the sampling time is $h = 1$ [18].

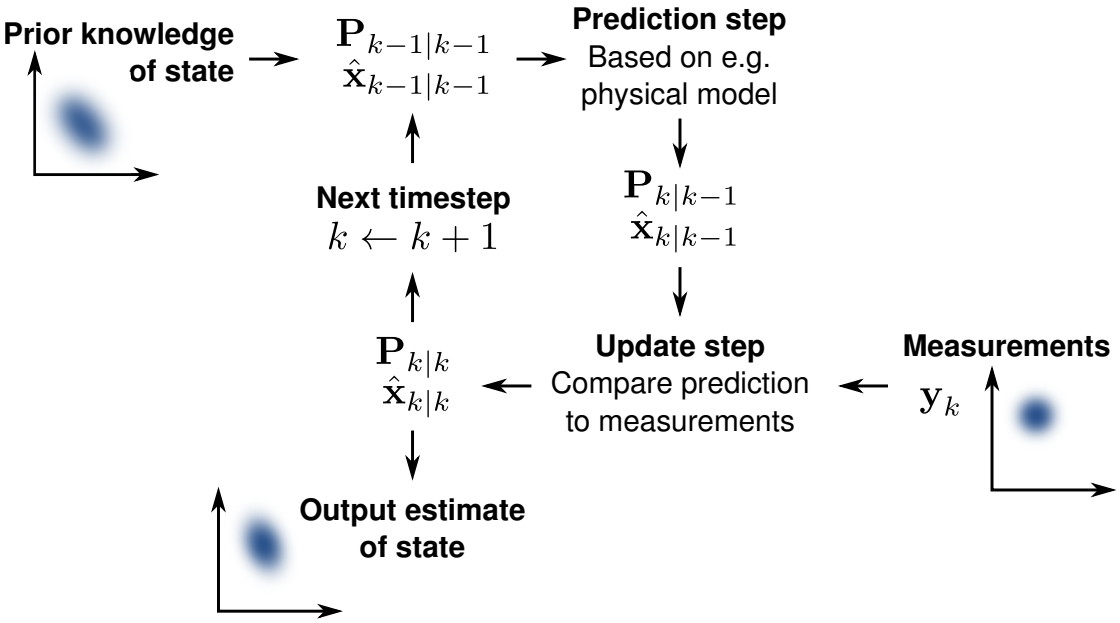

Figure 8.1: The Kalman filter keeps track of the estimated state of the system and the variance of the estimate. The estimate is updated using the state transition matrix, Eq. 27, and measurements. Source: Petteri Aimonen [1].

Predict

Predict an *a priori* state estimate through the discretized model of the system

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{\Phi}_k \hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{\Gamma}_k \boldsymbol{u}_{k-1}.$$

Predict an *a priori* covariance estimate using the previous covariance estimate and the current system and process covariance matrices

$$\mathbf{P}_{k|k-1} = \boldsymbol{\Phi}_k \mathbf{P}_{k-1|k-1} \boldsymbol{\Phi}_k^T + \mathbf{Q}_k.$$

Update

Compute the measurement residual using the current measurement and the current state estimate

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{y}_k - \mathbf{C}\hat{\boldsymbol{x}}_{k|k-1}.$$

Compute the residual covariance matrix from the observation matrix and the measurement noise covariance matrix

$$\mathbf{S}_k = \mathbf{C}\mathbf{P}_{k|k-1}\mathbf{C}^T + \mathbf{R}.$$

The optimal Kalman gain at time k is now

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{C}^T\mathbf{S}_k^{-1}.$$

Update the *a posteriori* state estimate using the Kalman gain and the innovation

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} - \mathbf{K}_k\tilde{\boldsymbol{y}}_k.$$

Update the *a posteriori* estimate covariance matrix using the Kalman gain and the *a priori* covariance estimate

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{P}_{k|k-1}.$$

8.2 Noise Specifications

Much process noise was expected from the odometer sensor due to e.g. gearbox backlash, friction, physical disturbances, etc. The gyroscope, however, was considered to be very accurate. As the first three states are a combination of the odometer and the gyroscope sensor data the resulting standard deviations had to be qualified guesses. The standard deviations for the last four states, with known noise characteristics, were given by the specifications for the gyroscope [15]. While the real dynamics of states four and five were not known, they were updated using transformed gyroscope sensor data which is why the standard deviations for these states were chosen from the gyroscope specifications. The values are presented in Table 8.1.

σ	**Value**	**Unit**
σ_{x_1}	0.1	m
σ_{x_2}	0.1	m
σ_{x_3}	0.1	m
σ_{x_4}	$1.5 \cdot 10^{-4}$	rad
σ_{x_5}	$1.5 \cdot 10^{-4}$	rad
σ_{x_6}	$5.6 \cdot 10^{-6}$	s
σ_{x_7}	0.001	s
σ_{x_8}	$5.6 \cdot 10^{-6}$	s
σ_{x_9}	0.001	s

table 8.1: Standard deviations for the process noise.

The GPS sensor was considered accurate with only a small amount of noise corruption and its position was considered to be the true position of the vehicle. Because of this, the measurement noise for the GPS was modeled as the noise of the noise i.e. how much the noise varied and not how much the measurement varied. The last available measurement, the vehicle heading, proved to always be unreliable and so its standard deviation was chosen to be large. The values, all of which are educated guesses, are presented in Table 8.2.

σ	**Value**	**Unit**
$\sigma_{x_{gps}}$	0.1	m
$\sigma_{y_{gps}}$	0.1	m
$\sigma_{z_{gps}}$	0.1	m
$\sigma_{heading}$	10	rad

table 8.2: Standard deviations for the measurement noise.

The weighting matrices for the Kalman filter are thus

$$\mathbf{R} = \mathcal{E}\{\boldsymbol{v}_k \boldsymbol{v}_k^T\} = \begin{bmatrix} \sigma_{x_{gps}}^2 & 0 & 0 & 0 \\ 0 & \sigma_{y_{gps}}^2 & 0 & 0 \\ 0 & 0 & \sigma_{z_{gps}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{heading}^2 \end{bmatrix}, \tag{42}$$

$$\mathbf{Q} = \mathcal{E}\{\boldsymbol{w}_k \boldsymbol{w}_k^T\} = \begin{bmatrix} \sigma_{x_1}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{x_2}^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{x_3}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{x_4}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{x_5}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{x_6}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{x_7}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{x_8}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{x_9}^2 \end{bmatrix}, \tag{43}$$

and

$$N = \mathcal{E}\{\boldsymbol{w}_k \boldsymbol{v}_k^T\} = 0, \tag{44}$$

as the noise vectors are assumed to be mutually independent.

9 Test Cases for the Real Vehicle

The tests on the real vehicle were all performed inside the Skaraborg Regiment P4 Armored Vehicle facility in Skövde, using various models of the CV90. The exception to this is the test described in the first subsection, which was a test conducted by BAE Systems before this thesis started.

Even though a lot of different tests were done, not all of them were used in the development of the algorithms or in the model verification process, and so only a subsection of them have been described in detail in the following subsections. The *Test* number in the lists represent the tests done during that particular day and this number resets every new test day. The *case* numbers is a running total of the tests but mainly functions as a way to keep track of the independent tests in MATLAB. The arrow that is present in all figures indicates the driving direction.

9.1 Previous Test

This test case was done in order to evaluate the performance of the NAV90 INS as well as to explore the performance of an possible INS that used the two-axis gyroscope system installed on some CV90 vehicles, see Fig. 9.1. The data from this test was used in the early stages of this thesis to test the algorithms.

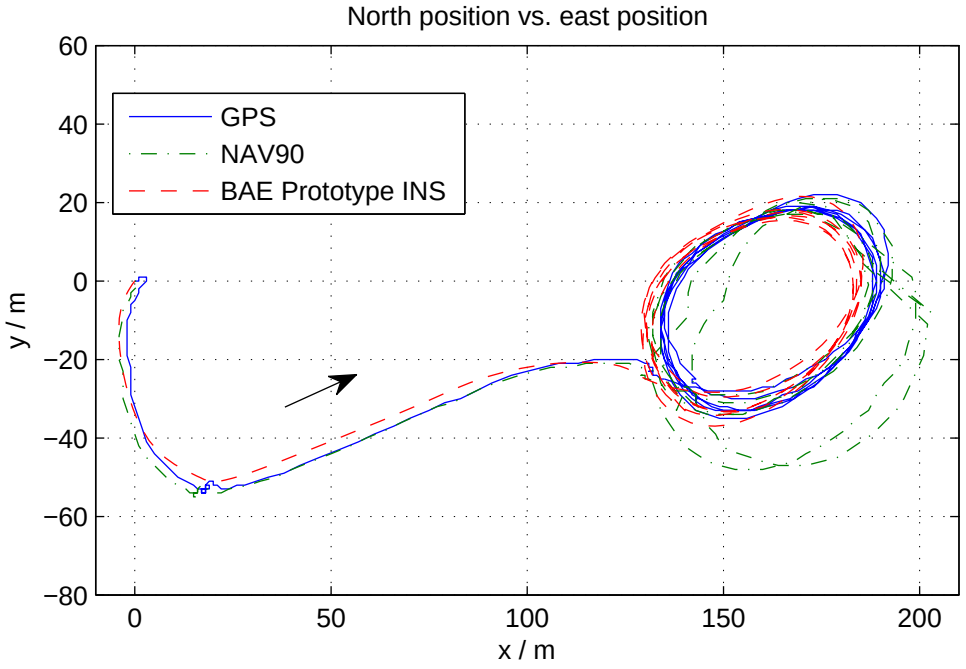

Figure 9.1: The previous test where the vehicle was driven on asphalt. The NAV90 system's erratic behavior is most obvious in the circular motions on the right hand side of the plot. Source: Erik Nygård [11].

9.2 Skövde, 2012-10-10

During this day 12 different tests were performed, 11 were done on a CV9040C and 1 was done on a CV9040TD. Due to complications with the GPS receiver, GPS data were not collected during any of the tests performed during this day. The fact that the GPS was not functioning made some of the planned tests, where the vehicle was to be moving along a track, unnecessary and so only tests where the vehicle was standing still were performed. All the tests are described further in Table 9.1.

Test (case)	Log ID	Description
1 (1)	092740	Standing still inside for a duration of 8 minutes.
2 (2)	093632	Standing still inside for a duration of 1 minute.
3 (3)	094848	Waited 10 minutes after the last test and then logged for 1 minute while the vehicle was standing still.
4 (4)	095916	Waited 10 minutes after the last test and then logged for 1 minute while the vehicle was standing still.
5 (5)	100228	Standing still inside for a duration of 8 minutes.
6 (6)	102754	Standing still outside for a duration of 2 minutes.
7 (7)	104052	Standing still outside for a duration of 4 minutes.
8 (8)	112907	Standing still outside for a duration of 4 minutes.
9 (9)	114247	Standing still outside while the turret rotated 2 complete turns. The elevation angle was locked at 0 degrees.
10 (10)	114706	Standing still outside while the turret rotated 2 complete turns. The elevation angle was locked at approximately 24 degrees.
11 (11)	114940	Standing still outside with the turret locked at 0 degrees. The weapon was raised and lowered two times between in an interval spanning between its maximum and minimum elevation angles.
12 (12)	133624	Standing still inside for a duration of 1 minute and 20 seconds.

table 9.1: table showing the various test that were done during the first test day in Skövde on 2012-10-10.

9.2.1 Test 8 (8)

This test was used in order to find any offset present in the stationary gyroscope signals. These signals were averaged and it was found that the offset in each channel was 0.52 mrad/s around the z-axis, i.e. yaw, and 0.08 mrad/s around the y-axis, i.e. pitch. These values were then subtracted from the gyroscope signals in the algorithms to improve the performance of the simulations, since the majority of the simulations modeled the data of the exact same vehicle.

9.2.2 Test 9 (9) and Test 10 (10)

A comparison between the carrier yaw rotation during these two test was performed in order to find out how the angle of the weapon affected the heading estimate. Fig. 9.2 shows the yaw angles for both of these tests.

Figure 9.2: Two tests where the turret performed two full rotations with the weapon at different elevation angles. A loss in heading estimate accuracy was observed when the weapon elevation angle was significantly different from the original 0° position.

It is apparent that when the weapon is in an angle significantly different from the default rest position, the heading estimate deteriorates. This shows the limitation that is inherent in using a two-axis gyroscope in a system which has three degrees of freedom.

9.3 Skövde, 2012-10-26

The tests done during this day focused on collecting data while the vehicle was moving, since this had not been possible during the previous test day. The primary course for the vehicle was decided to be a square-like shape, and different dynamics of the turret and weapon were tested during each run, see Table 9.2.

Test (case)	Log ID	Description
1 (13)	103916	The vehicle was driven along the square track with the weapon and turret locked at 0 degrees, i.e. pointing straight ahead in relation to the carrier.
2 (14)	104230	Same track as the previous test with the exception of idling the vehicle ca 1 minute before and after driving the track. This was to check so that the algorithm could handle the case where the system was fixed in position, i.e. when the velocity was 0 m/s.
3 (15)	105332	The vehicle was driven along the square track with the weapon stabilized at ca 0 degrees in elevation while the turret was pointed towards the center of the square for the duration of the test.
4 (16)	105907	The same test as the previous except the elevation angle was set to be close to maximum, i.e. around 24 degrees.
5 (17)	110315	A 2 minute test where the vehicle and all of its components were motionless. This test case was more of a comparison test where the data were compared with the similar test done during the previous test day.
6 (18)	110539	The vehicle had zero speed while the turret spun around 2 full rotations and the weapon was raised and lowered in a wavelike pattern.

table 9.2: table showing the various test that were done during the second test day in Skövde on 2012-10-26.

9.3.1 Test 2 (14)

The test was mostly used for verification of the model and it was noticed that the vehicle sometimes changed orientation of the NAV90 z-axis, sometimes it was ”up” and sometimes it was ”down”. It was also noticed that the initial idling time made the NAV90 able to find a good estimate of the initial heading, in fact it was the best heading estimate of all the tests done during this day.

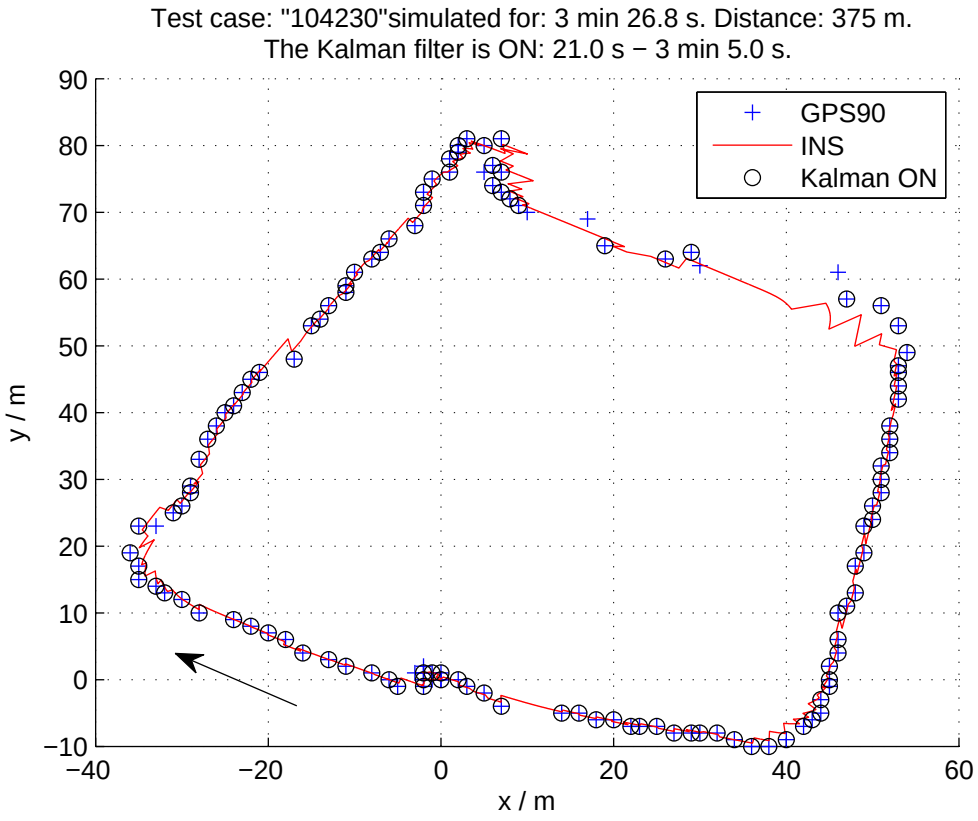

Figure 9.3: Position estimate with the INS assisted by the Kalman filter. Notice how the INS jumps back towards the GPS90 position and how the heading is not updated.

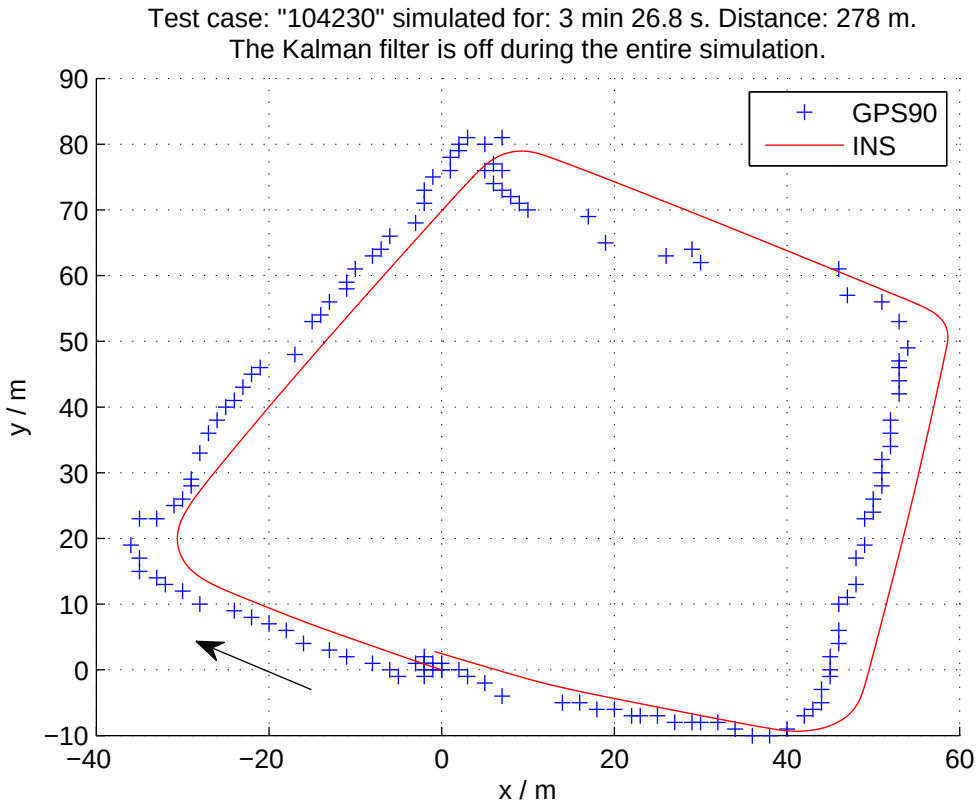

Figure 9.4: Position estimate with only the INS active. The path is much smoother since no jumping occurs. With a proper initial heading it is evident that the INS estimates the position well.

9.4 Skövde, 2012-11-15

This was the final data gathering excursion and also the one where the longest tests were run. The tests were done on the training grounds outside the P4 regiment base and the details are described in Table 9.3.

Data were lost during Test 3 (23) due to a bad connection and so this test will not be analyzed in detail. A laser was used to estimate distance towards a known waypoint, specifically the *TV Mast* waypoint located outside of Skövde proper, in order to evaluate the heading error and how this would affect the estimated location of targets. A glitch in the transfer of the laser ranging flag meant that the distance information could only be used on Test 1 (19, 20) and Test 5 (25, 26).

All plots have been reset so that coordinate $[0, 0]$ represents the *Amundtorp* waypoint. The green line in the plots represent the u-blox GPS coordinates and these are taken to be the ”true” coordinates, and so the distances to waypoints etc. are calculated using the u-blox GPS information. Furthermore, due to difficulties in synchronizing the NAV90 and the u-blox GPS UTC times, the u-blox data is slightly shifted to the right in the heading plots.

Test (case)	Log ID	Description
1 (19, 20)	115111 120137	Driving in hilly terrain and alongside a forest for the latter half of the circuit. Laser fired at the end. Laser: 6473 m (TV Mast).
2 (21, 22)	133817 134943	A long circuit from Amundtorp in the north to the firing range in the south. GPS data were unreliable during this test and data from the NAV90 system was lost during the final segment of the circuit.
3 (23)	140624	The vehicle moved forward ca 20 m, then reversed a few meters, then performed a center turn, and then finally went back to the starting position. The data connection was lost during most of the test and so nothing of value could be extracted.
4 (24)	141024	Same as Test 3 (23) except the data connection was good during the entire test.
5 (25, 26)	143146 144039	A long test taking the vehicle from the firing range in the south to a position northwest of Amundtorp in the north. The weapon was stabilized during the entire test and the laser was fired near Amundtorp. Laser: 6468 m.
6 (27, 28)	145530 150318	Same track as Test 1 (19, 20) but with a stabilized weapon and all combat hatches open. This was done in order to simulate a worst case scenario for the compass and the gyroscope.

table 9.3: table showing the various test that were done during the third and final test day in Skövde on 2012-11-15.

9.4.1 Test 1 (19, 20)

The track is followed well both with and without GPS assistance and the heading at the time of the range estimation to the TV Mast waypoint is very accurate. The final stage of the track for the two simulations can be viewed in more detail in Fig. 9.7 and Fig. 9.8, respectively. The norm of the error in the x and y coordinates of the position estimate, for the model with the Kalman filter activated, can be seen in Fig. 9.7. The error is smaller after the log switch mostly due to the fact that the vehicle is stationary for a short period during the end of the test. This could also explain why there is such a large difference between the u-blox GPS heading and the INS heading during the last part of the test.

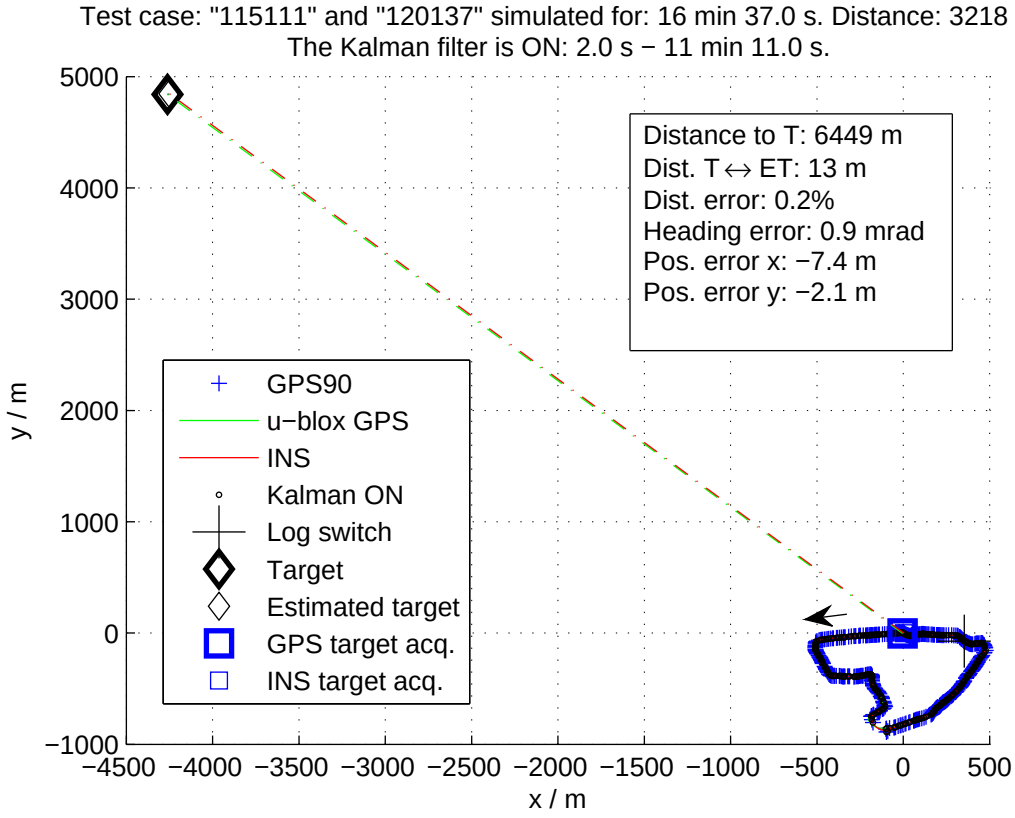

Figure 9.5: The circuit starts and ends approximately at the Amundtorp waypoint. With the Kalman filter on the heading error is only 0.9 mrad and the distance error is only 0.2%. The difference between the estimated ranging location, the small square, and the true ranging position, the large square, is $(-7.4, -2.1)$ m.

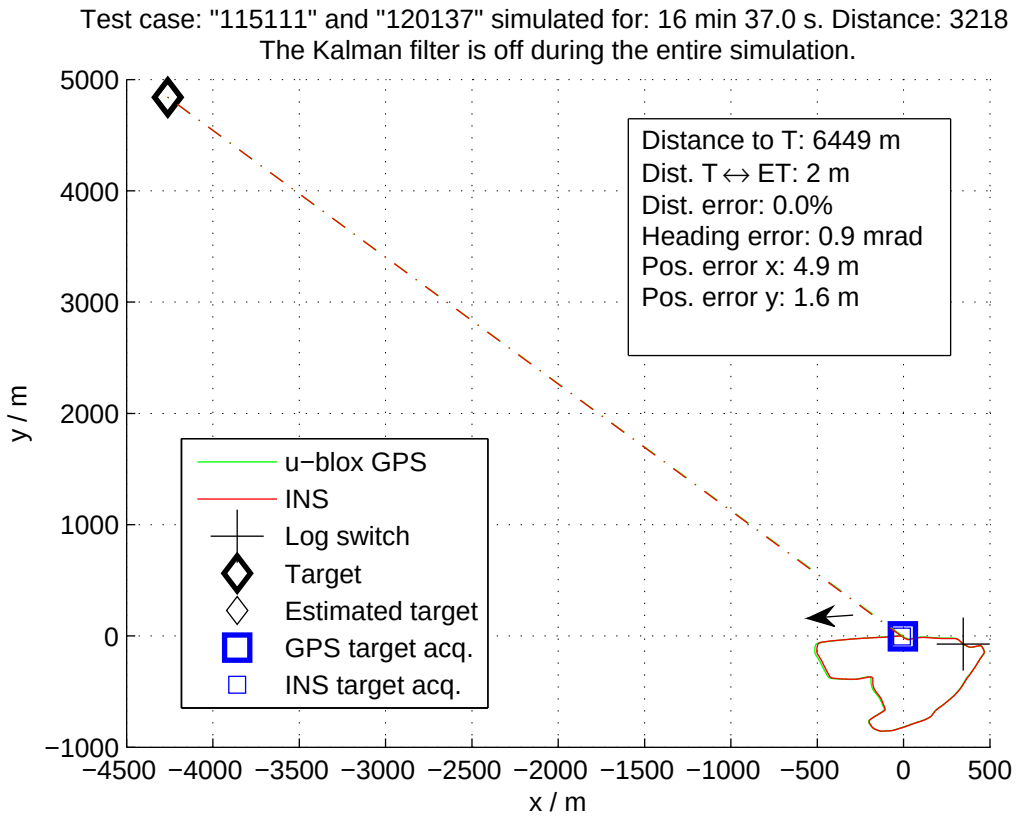

Figure 9.6: The circuit starts and ends approximately at the Amundtorp waypoint. With the Kalman filter off the heading error is still only 0.9 mrad but the distance error is 0.0%. The difference between the estimated ranging location, the small square, and the true ranging position, the large square, is $(4.9, 1.6)$ m. The smaller error, compared to the end difference for the GPS assisted INS coordinates, is probably due to the fact that the INS is pushed towards the GPS90 coordinates which are offset from the ”true” value of the u-blox GPS coordinates.

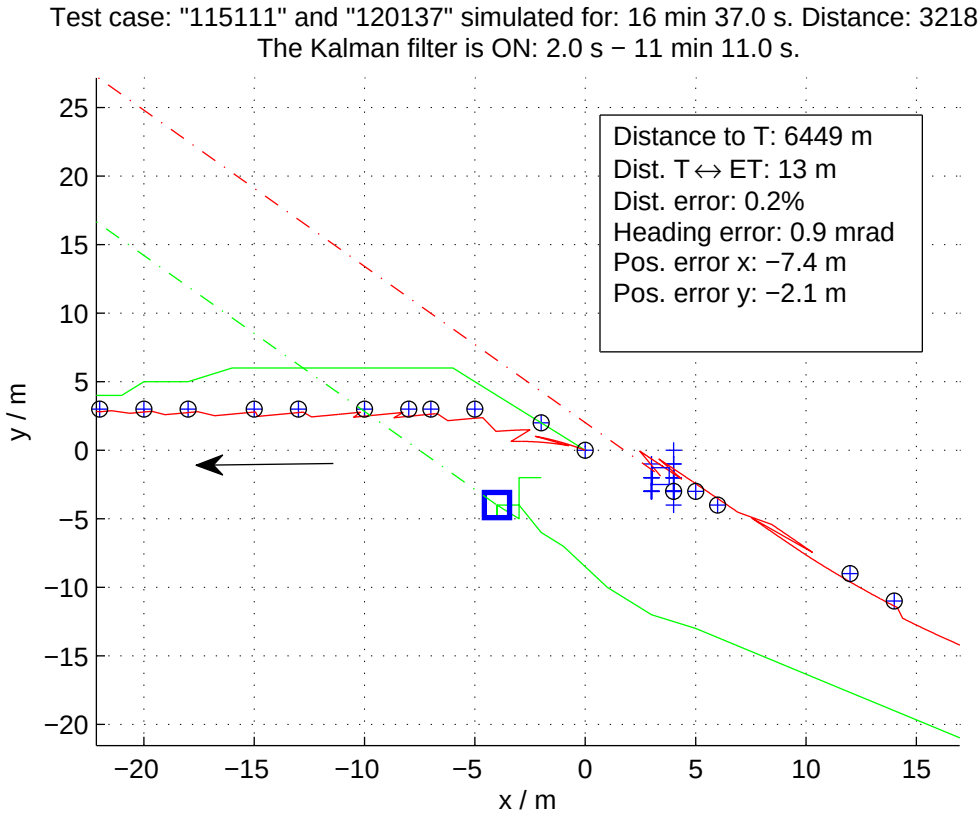

Figure 9.7: Since the u-blox GPS is taken as ”true”, the real laser ranging position is at the end of the green line, and not the last blue GPS90 coordinates that the INS relies on.

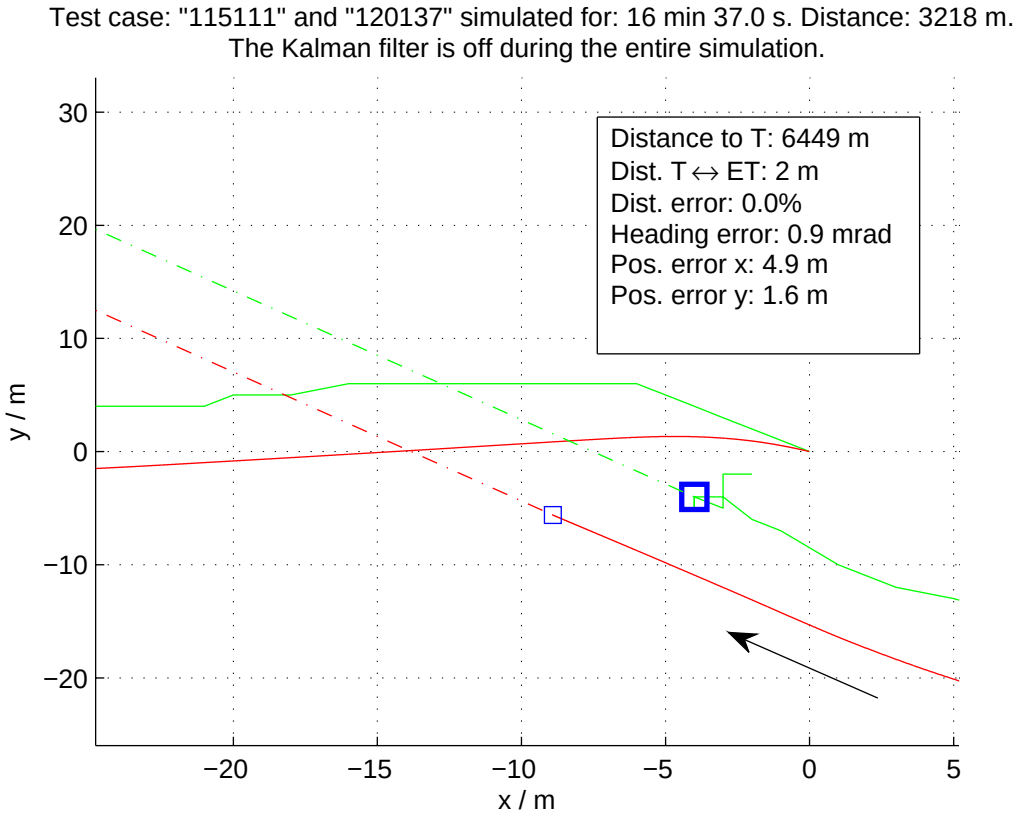

Figure 9.8: When no new information from GPS90 is relayed to the INS the final position is closer to the ”true” position of the u-blox GPS. This indicates that with a better GPS to depend on the INS could perform better.

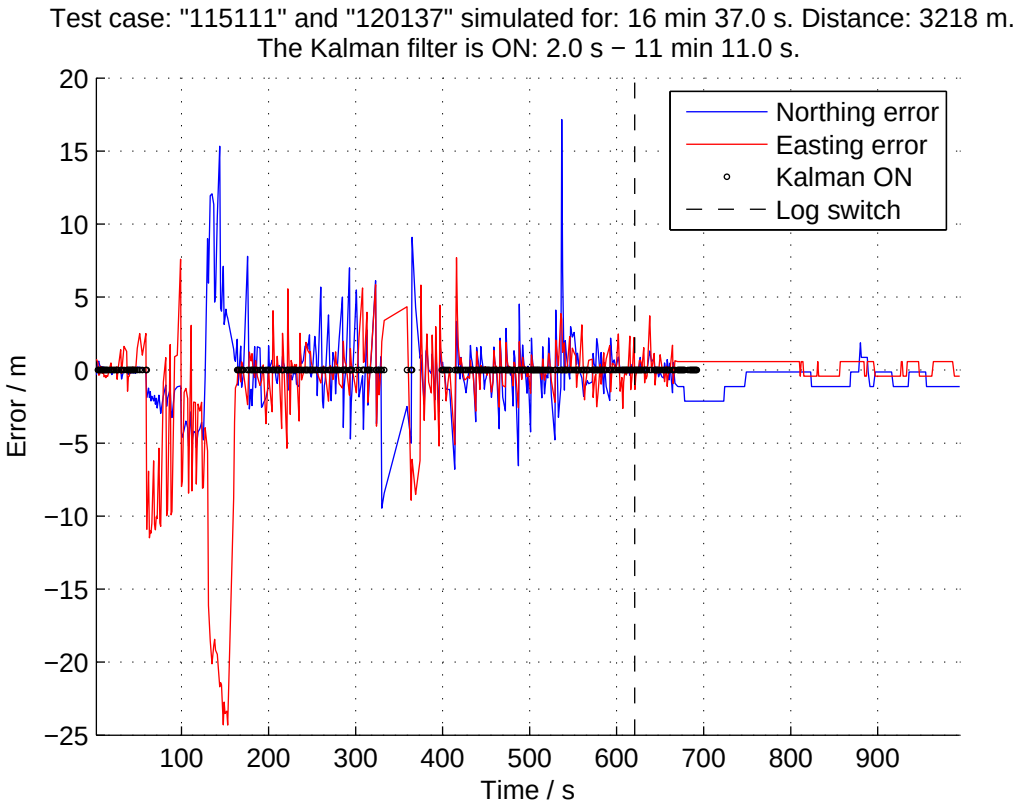

Figure 9.9: The error is relatively stable before the log switch, but it has a few spikes. After the log switch the INS heading is replaced with another value which gives a smaller error which is more stable since the vehicle is standing still and the Kalman filter does not reset the position.

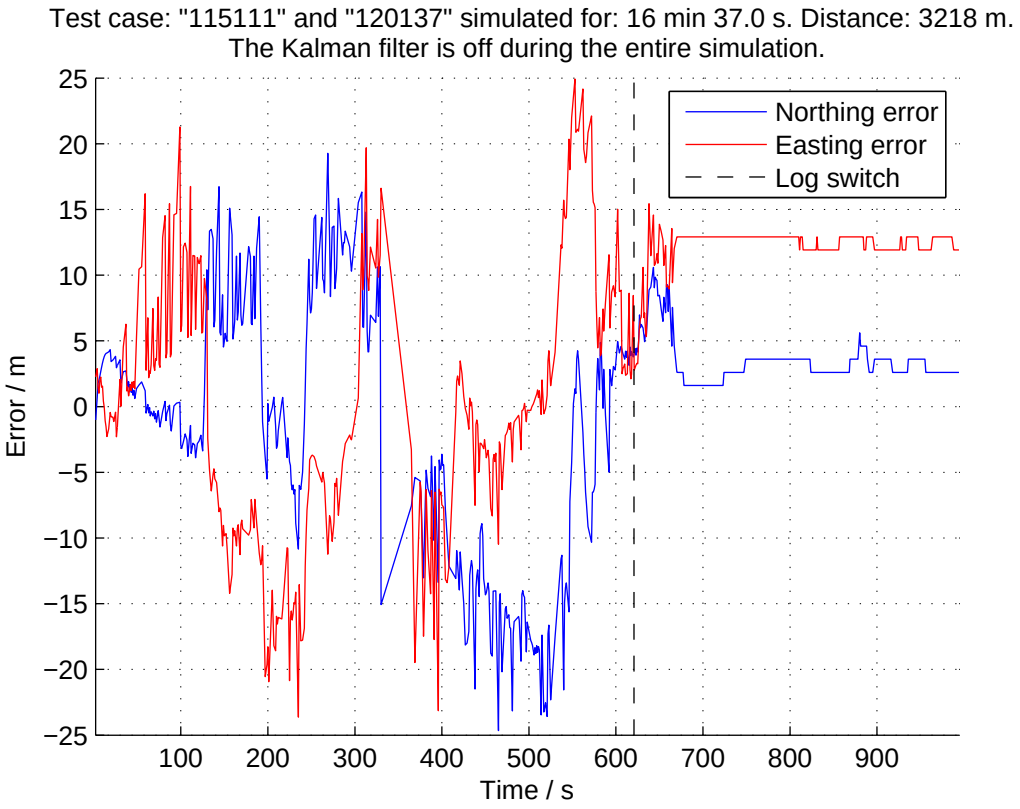

Figure 9.10: The error oscillates in the interval $[-25, 25]$ m and after the log switch, when the vehicle stops moving, it becomes stationary. Notice that the error shown specifies the difference between the INS and the GPS90 data, and not the more reliable u-blox GPS.

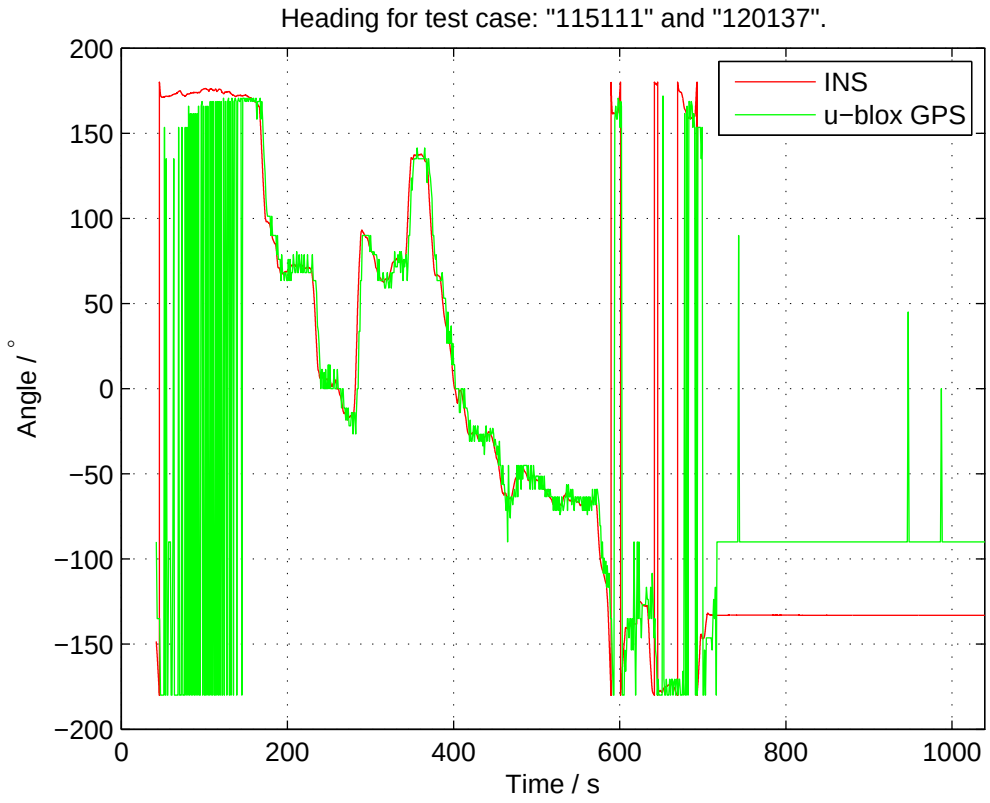

Figure 9.11: The u-blox GPS heading is only being sampled with $h = 1$ s which gives it a noisy appearance in the beginning. The INS estimates the heading well except in the end, however, during this part it is probably the u-blox which gives the wrong heading since the vehicle is stationary and the GPS coordinates are in the area close to that position. Note that the heading for both versions of the INS is basically the same since the heading is not affected by the GPS coordinates.

9.4.2 Test 2 (21, 22)

The track is followed quite well during the first part of the track, however, due to a faulty connection with the vehicle data bus, information was missing during a period at the end of the track. For this reason the last part has been moved to where it was estimated to have occurred if complete signal data were available, and it seems to fit relatively well. This test showed that the extracted data could be incoherent, and if that were the case the NAV90 system would not point this out via e.g. time stamps.

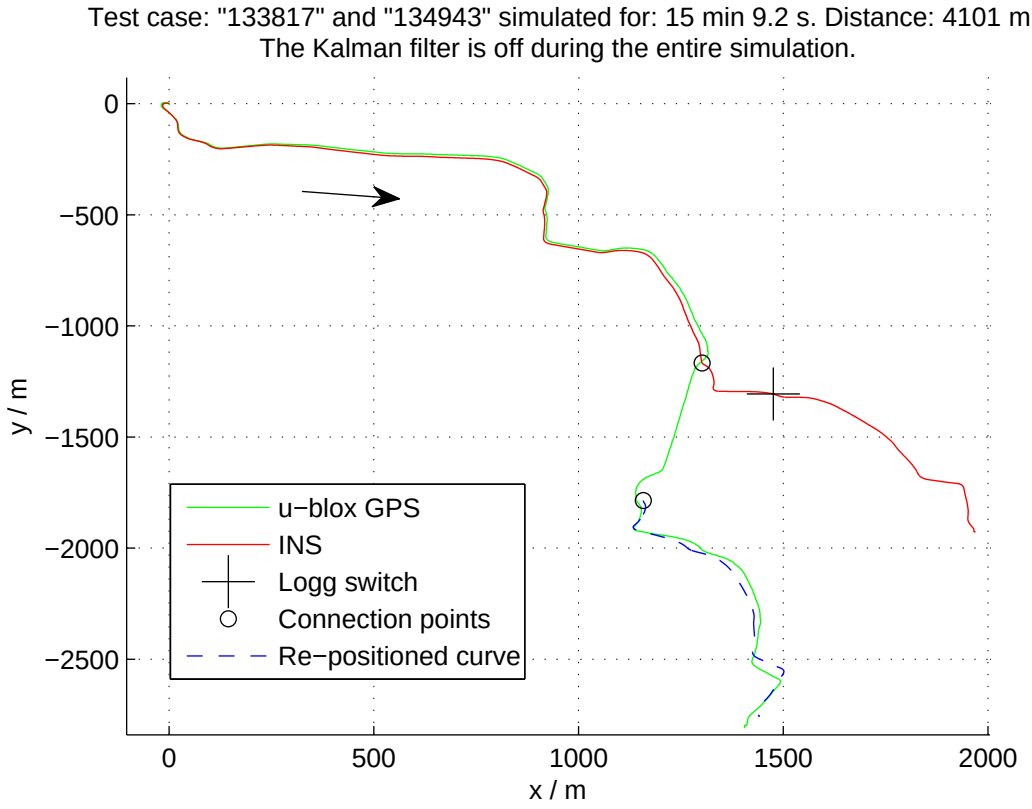

Figure 9.12: The connection points show where the original INS line has been cut in order to place it where it was estimated to have occurred during the actual test. The re-positioned curve seems to fit well but due to the loss of some information it is not a perfect fit.

9.4.3 Test 4 (24)

This test was done in order to find out how the NAV90 system relayed information regarding the speed of the vehicle when the vehicle turned and was in neutral gear, and when it reversed and was in reverse gear. NAV90 handled this by giving a speed of 0 km/h when the gear was in neutral and a negative speed when the gear was in reverse. This was fortunate since this meant that no ”gear indication flag” had to be extracted from the carrier data bus in order to help the algorithm know if the vehicle moved forwards or backwards.

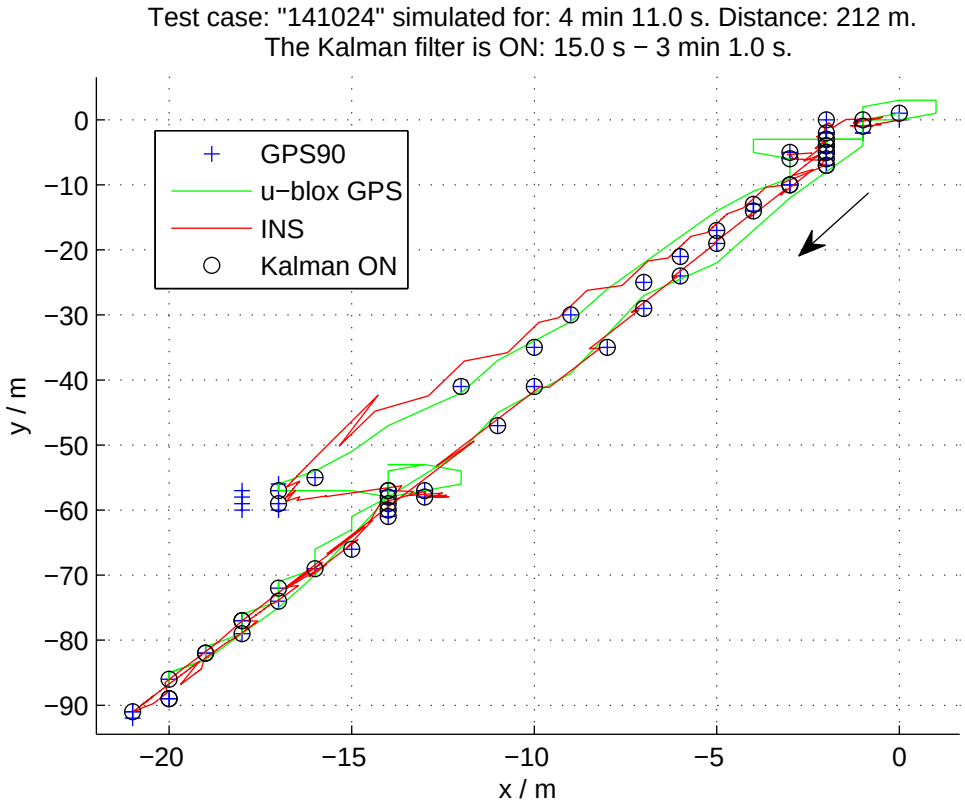

Figure 9.13: The INS follows relatively well since it is moved each time a new GPS value is obtained.

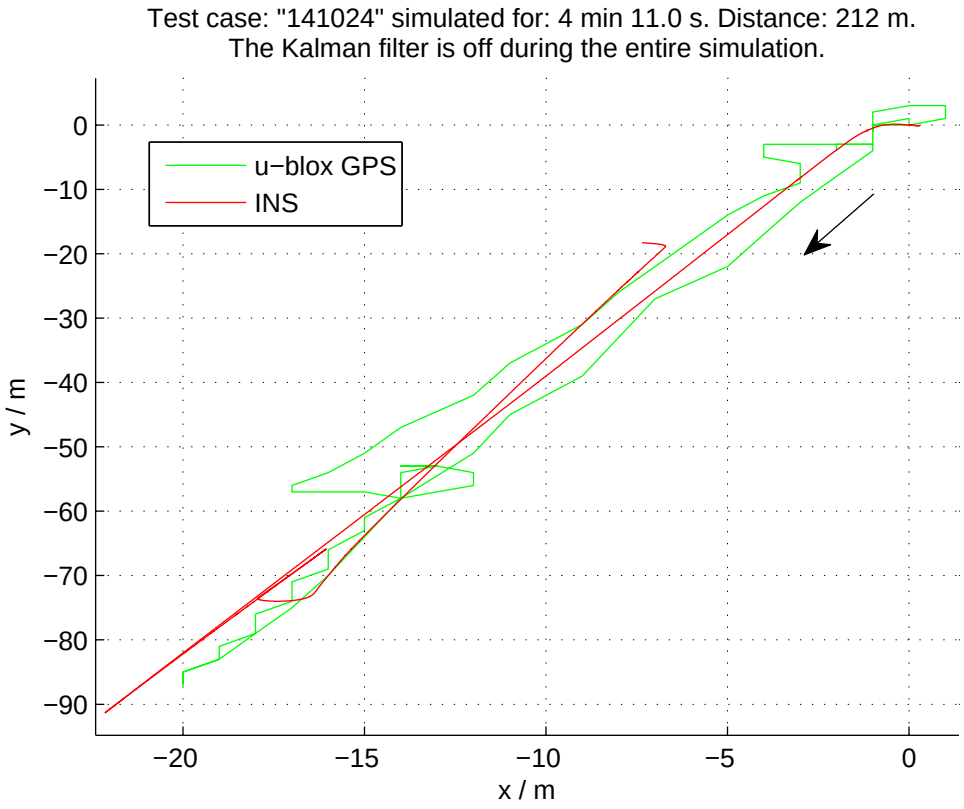

Figure 9.14: The INS does not come all the way back after the vehicle has performed the carrier rotation during the middle part of the test, in the lower left hand side of the plot, the reason for this is unknown.

9.4.4 Test 5 (25, 26)

The true position on the track was estimated well in both cases and the low heading error shows that the INS followed the true heading accurately. The log switch did not introduce any major deviations from the real target position. The error is stable for the entire test except for a spike that occurs at the log switch.

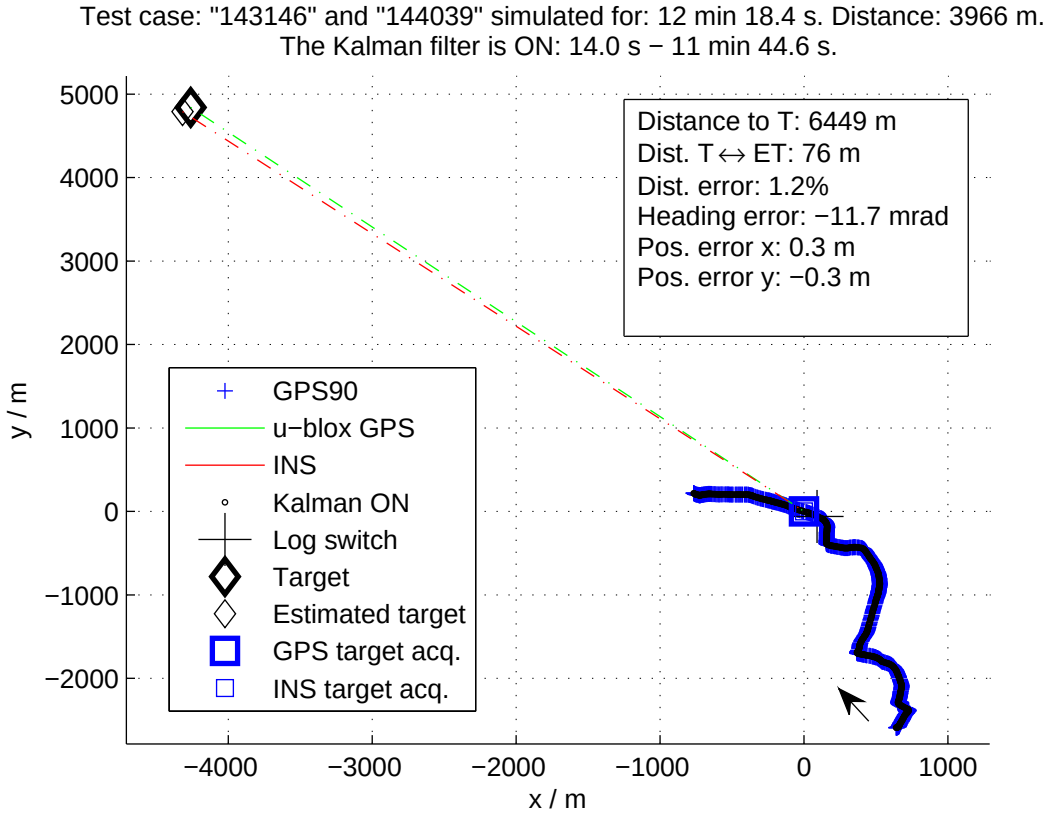

Figure 9.15: The laser ranging occurred at the Amundtorp waypoint. The heading error was relatively high compared to Test 1 (19, 20), but still within acceptable boundaries. The same could be said for the distance error. The difference between the estimated ranging location, the small square, and the true ranging position, the large square, is $(0.3, -0.3)$ m.

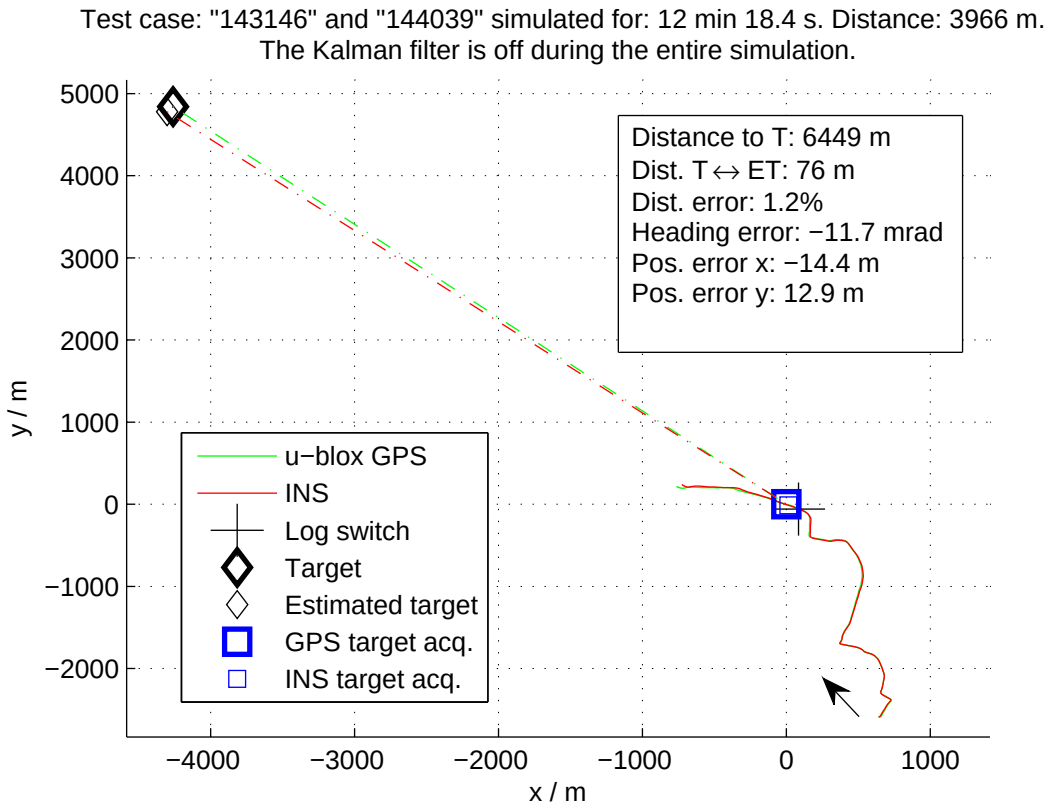

Figure 9.16: The laser ranging occurred at the Amundtorp waypoint. The heading error was the same as for the simulation with the Kalman filter but the distance error was slightly larger, probably due to the fact that the INS curve no longer ”jumped” toward GPS90 coordinates. The difference between the estimated ranging location, the small square, and the true ranging position, the large square, is $(-14.4, 12.9)$ m. The larger error, compared to the end difference for the GPS assisted INS coordinates, is probably due to the log switch and the change it creates in the system.

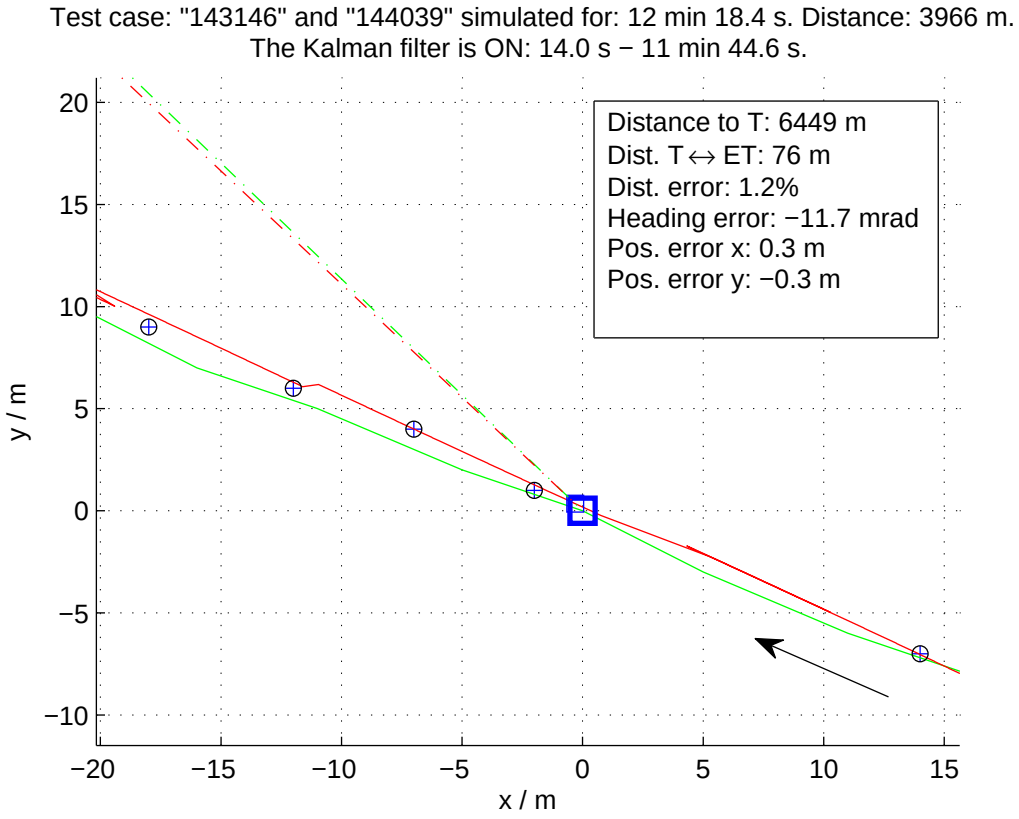

Figure 9.17: A detailed view of the laser ranging position. The estimated ranging position occurs a few meters behind the true position. The reason for this could be a synchronization issue between the log files since the ranging happens just moments after a log switch had taken place.

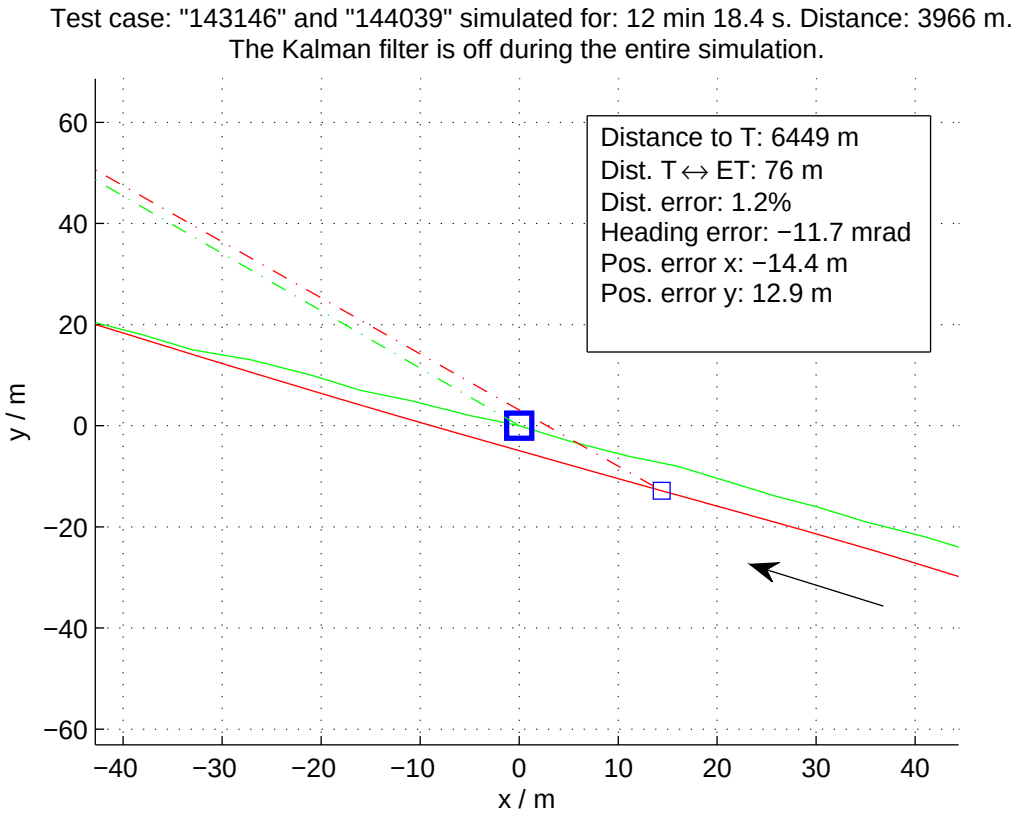

Figure 9.18: Same heading and distance error as the simulation with the Kalman filter, but since the position is not pushed along, as it is in the case with the filter, the final position error is slightly larger.

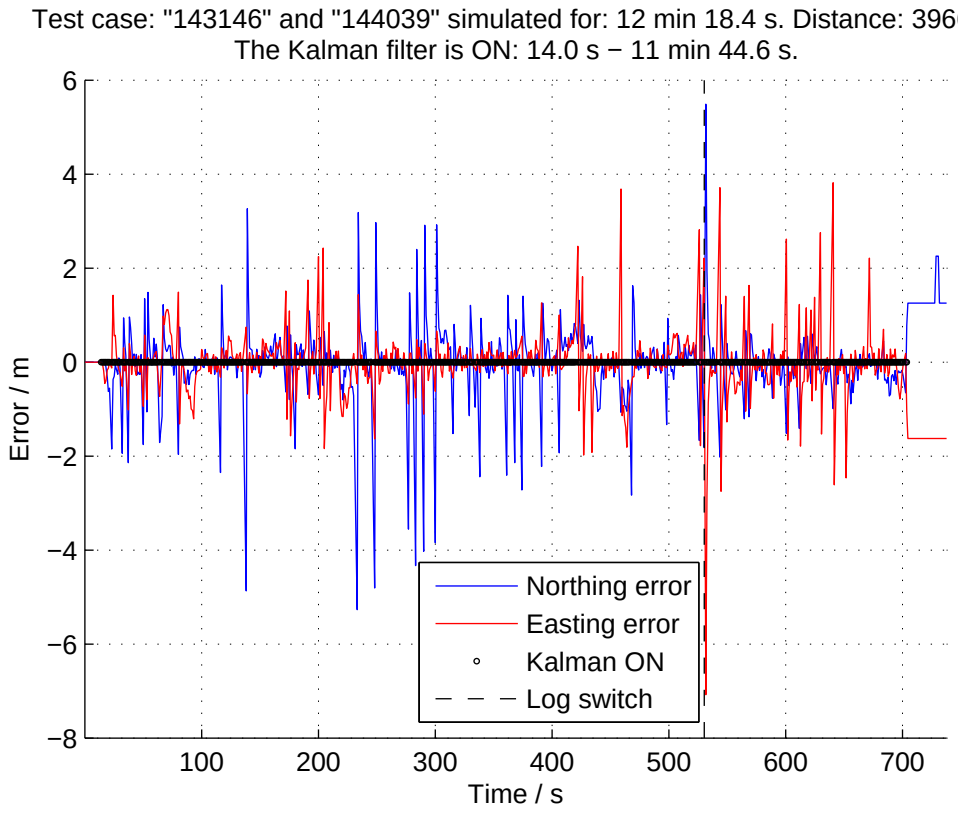

Figure 9.19: The error is stable both before and after the log switch. However, a spike occurs at the log switch and the reason for this could be a synchronization issue.

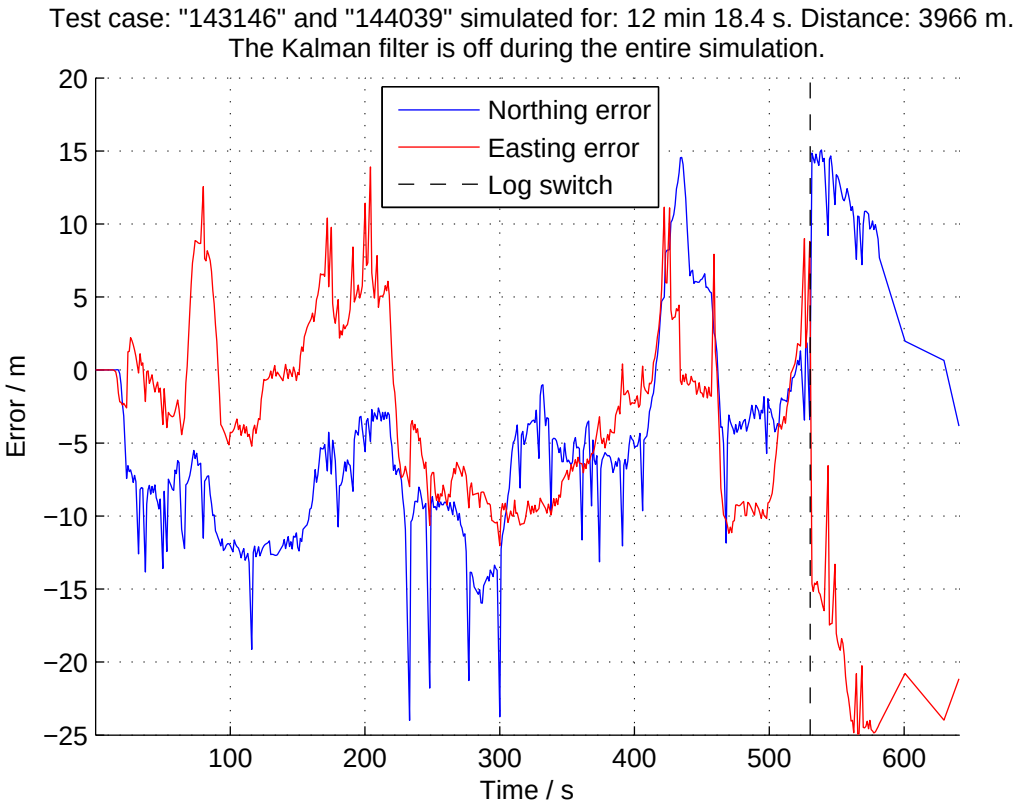

Figure 9.20: The error oscillates in a slightly smaller interval compared to Test 1 (19, 20), Sec. 9.4.1, and the oscillation has a longer period. This could be due to the straighter path, compared with the somewhat circular path performed in Test 1 (19, 20), that the vehicle is traveling during this test. Notice that the error shown specifies the difference between the INS and the GPS90 data, and not the more reliable u-blox GPS.

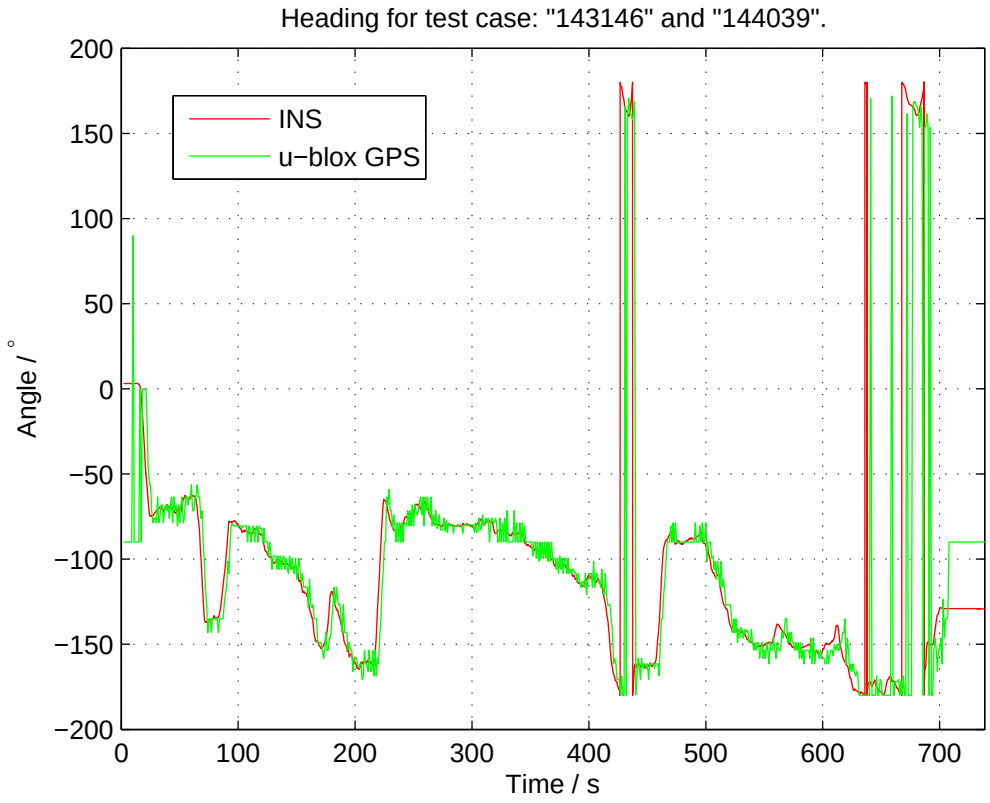

Figure 9.21: The heading corresponds well to the somewhat noisy u-blox GPS heading. The idling times in the beginning and the end of the tests are apparent in the plot in the form of large differences between the headings. Note that the heading for both versions of the INS is basically the same since the heading is not affected by the GPS coordinates.

9.4.5 Test 6 (27, 28)

This test was designed to really strain both the NAV90 system, by having all the battle hatches open to interfere with the compass, as well as the gyroscope, by constantly rotating the turret and moving the weapon.

It was expected that the GPS assisted system would handle the test well, and it did, but also the INS without GPS handled the challenge surprisingly well. It was surprising because in previous tests on manually generated test tracks the INS had not given results of this caliber, but this could also be due to the fact that it is difficult to recreate real world signals.

It is noticeable, however, that after the log switch the INS travels a shorter path than the real system. The explanation for this could be that during the log switch the system believed it had a small pitch angle, i.e. moving along an upwards slope, and during the down time of the log switch, which was 4 s, the vehicle could have returned to traveling parallel to the ground. The INS would not have integrated this signal since it did not receive information during the down time and would have continued to believe that the vehicle was traveling upwards.

Since the plots show the coordinates projected in the xy-plane, the projected path would be shorter. This suspicion was verified when the data showed that the system believed that it was at a height of 400 m at the end of the track, when in reality it should have been at 0 m. This shows that the INS can be improved in this area and it is discussed further in Sec. 10.

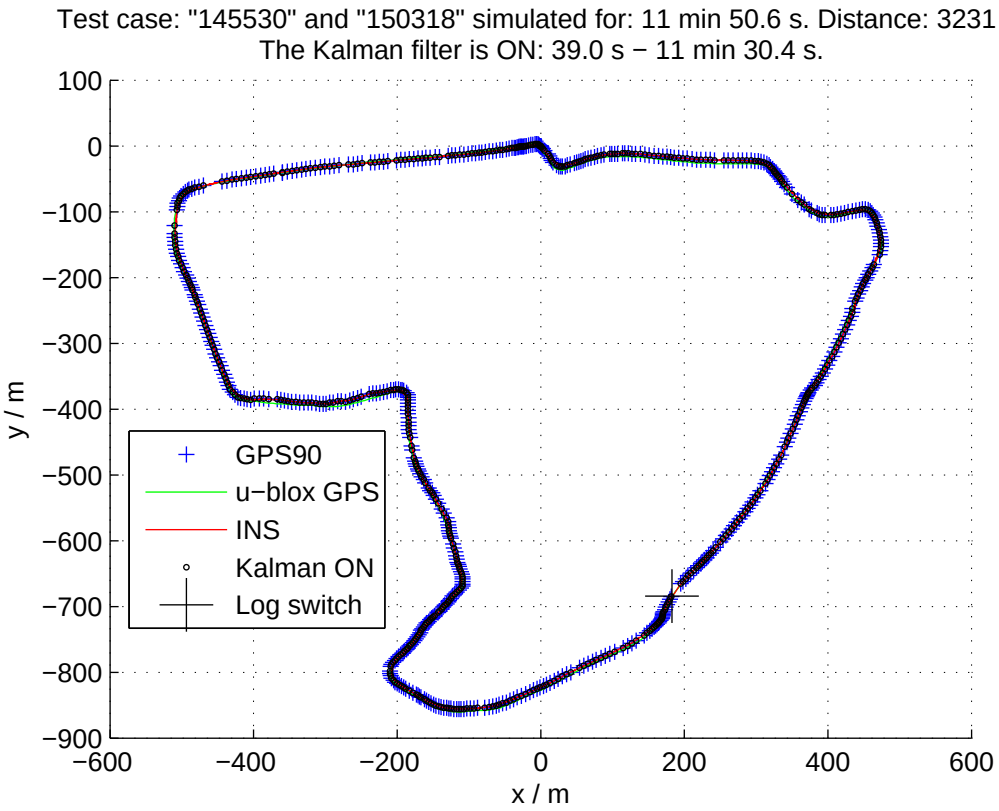

Figure 9.22: As expected the track is followed well when the GPS is activated.

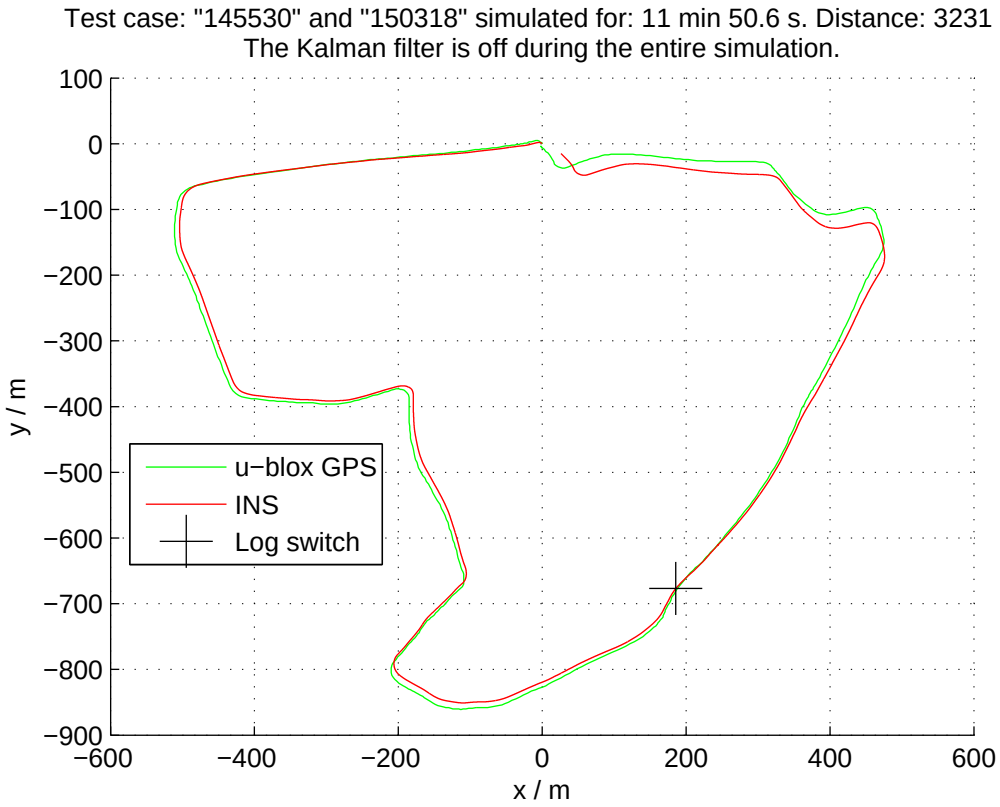

Figure 9.23: The track is followed with surprising accuracy when no GPS data is available. However something happens at the log switch and the system believes it is traveling with a non-zero pitch angle for the rest of the test, which results in the somewhat shorter path.

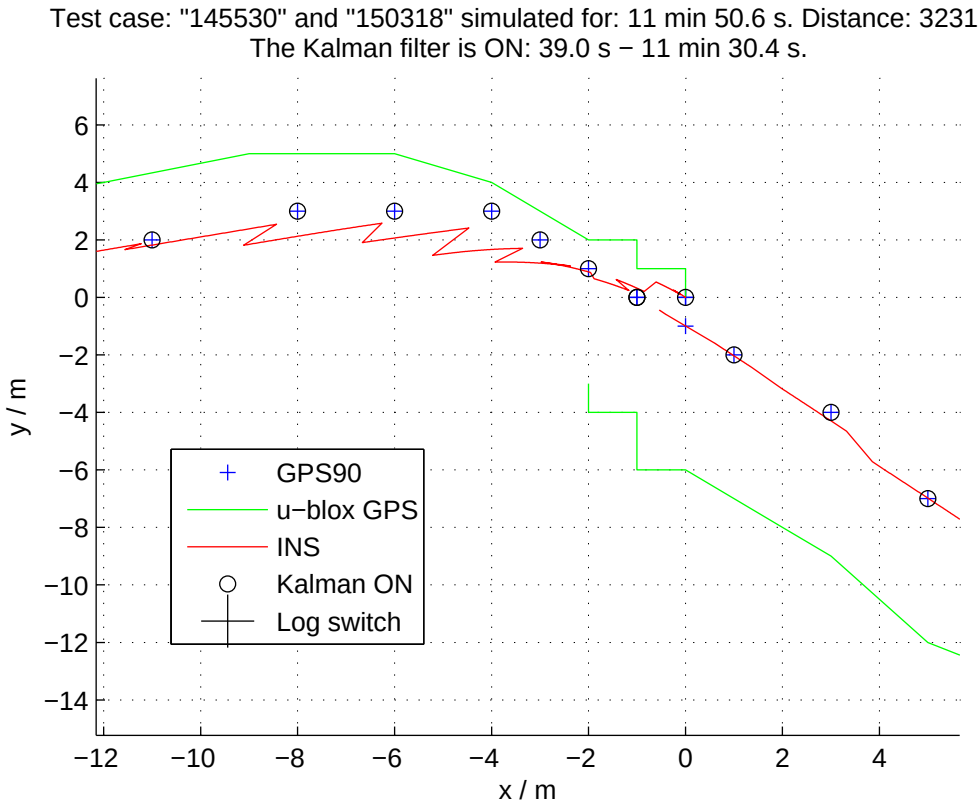

Figure 9.24: The system functions as expected and has an accurate heading in the last part of the track.

Figure 9.25: The shift in between the INS and u-blox GPS position is evident and it results in large end position errors.

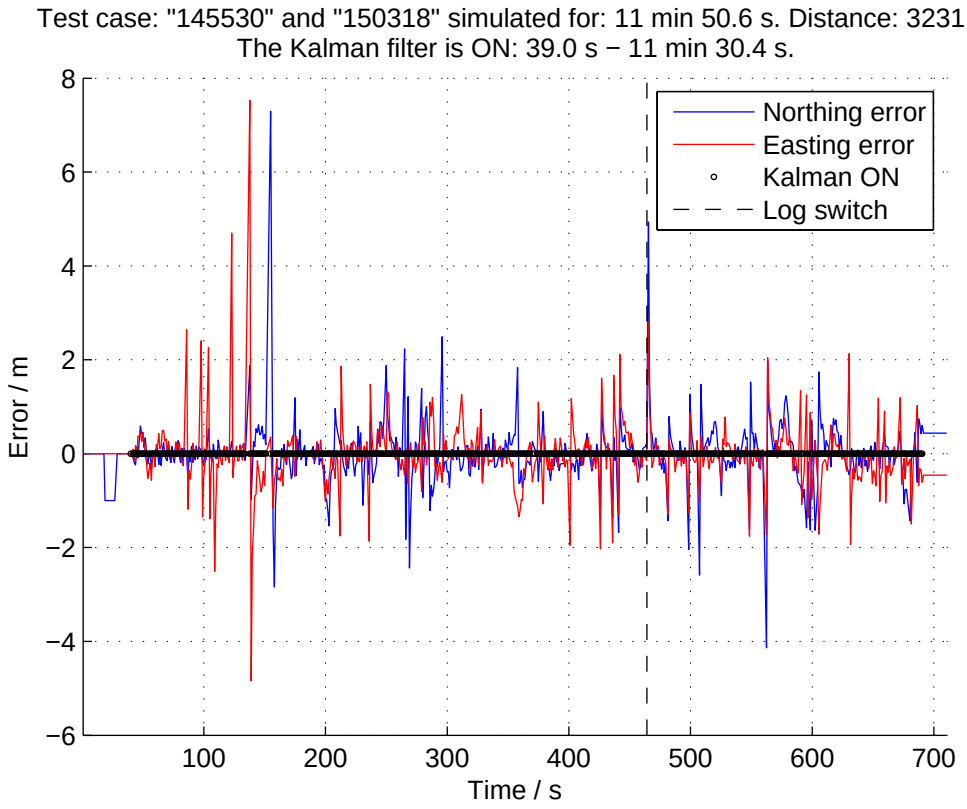

Figure 9.26: The error is relatively stable with the exception of a few spikes. No noticeable differences can be seen after the log switch.

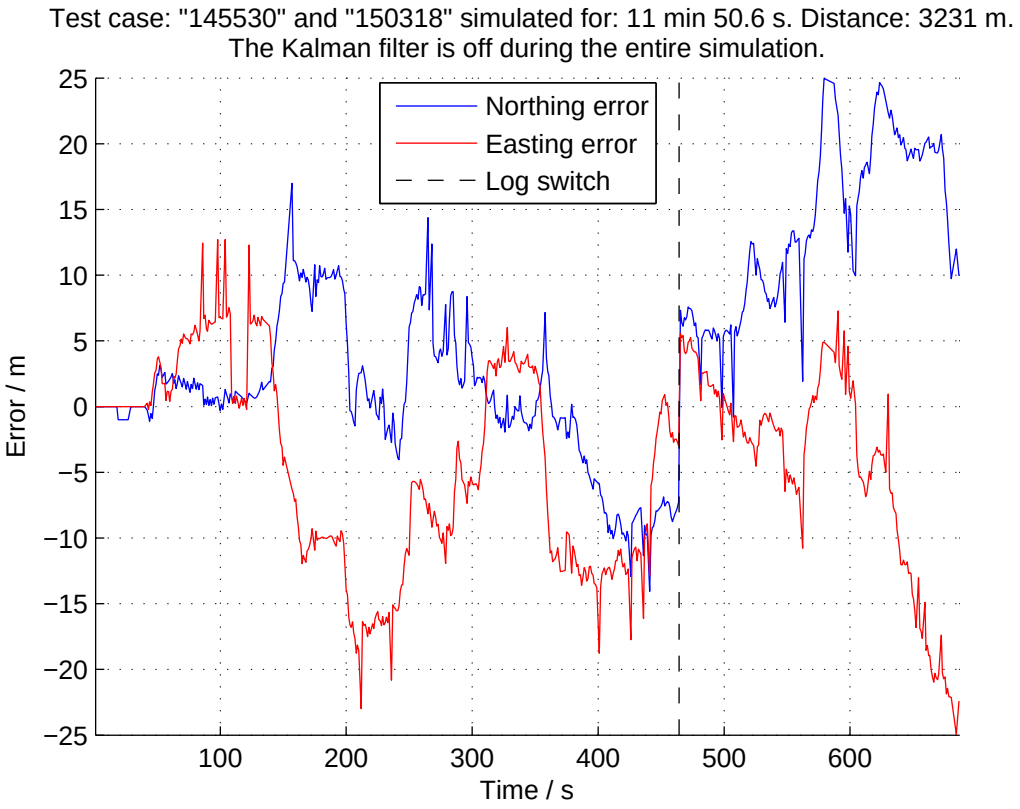

Figure 9.27: The error magnitude is relatively stable before the log switch, with a magnitude only reaching up to 25 m. After the log switch, however, the error begins to diverge. This is probably because of the INS thinks that the vehicle has a negative pitch angle, i.e. that the vehicle is following an upward-pointing velocity vector.

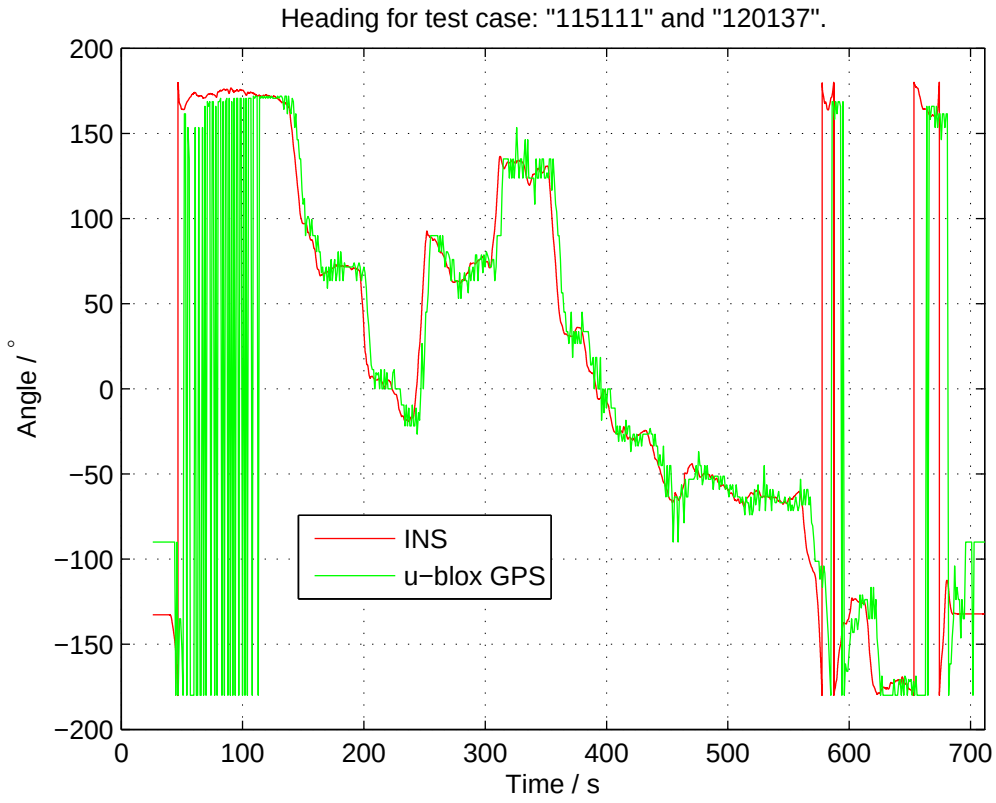

Figure 9.28: The heading is followed very accurately except in the beginning and end of the track when the vehicle is stationary. The synchronization issue between the heading logs causes a shift which is apparent in the plot. Note that the heading for both versions of the INS is basically the same since the heading is not affected by the GPS coordinates.

10 Problems that were Encountered

The data extraction from the vehicle had some problems associated with it. The extraction was slowed down by data logs that did not follow any predetermined pattern for saving data and because of this they had to be parsed and rearranged in Matlab to follow a standard layout.

Important data such as PFP flags, which were used for finding the time stamps at which laser ranging had taken place, were not always extracted properly which resulted in that a majority of the tests lacked any ranging information.

The time limit that imposed restrictions on the amount of data that could be collected in a single log file produced gaps in the simulation that were hard to fix, and the NAV90 system gave no indication, via e.g. time stamps, when said gaps had occurred in the logs.

At one occasion the extraction tool started to overwrite the current log file from the beginning, in all columns except the time column. This caused the beginning of the test to become unavailable and at the same time it was hard to find where the newly overwritten data ended and the real data started. Due to this a section of that test had to be cut out before simulation.

Further problems were encountered with the GPS90 system, which is the current GPS receiver on the majority of CV vehicles. The system had a long startup time, sometimes not starting at all, and the data it produced had relatively low resolution. The data also had many outliers which, if they were not removed or filtered, introduced errors in the simulations.

Another problem with the mutual dependence of the GPS90 and NAV90 systems was the fact that if GPS data were unavailable, then the NAV90 UTC time did not function and this made it impossible to sync NAV90 logs to other logs via UTC time.

The biggest problem was that the third gyroscope, which was installed on the third test day, could not be used since the flag responsible for syncing the third gyroscope's log with the NAV90 log could not be extracted properly due to a software bug. In the beginning of the project this was an important part of the evaluation of the new INS, to see if it could be significantly improved with a third gyroscope axis. This part of the thesis had to be abandoned due to the problems with the data extraction.

A problem that may appear in the future development of a new INS that uses this thesis as a basis is rewriting the various algorithms and functions to a fast and efficient program intended for embedded systems. The modeling and simulations were mainly performed within the Matlab and Simulink environments and some of the functions were CPU intensive, e.g. transformations. Among the biggest concern for an embedded implementation is the system matrix discretization which requires the matrix exponential to be recomputed each time step.

11 Conclusions and Recommendations

This thesis has shown that with the right tools it is possible to introduce an inertial navigation system for the CV90 platform that utilizes the two-axis gyroscope present on most CV90 vehicles. The performance of this INS has been deemed to be excellent and it is a worthy successor to the previous NAV90 system. The INS can estimate both position and heading in a robust manner for time periods in excess of 10 minutes, which can be considered to be a long duration since the standard operating times for an INS on this platform is active only a few minutes at a time in worst case scenarios.

It is clear from the testing and validation of the algorithms that the new INS can work in real world scenarios. However, there is room for improvement.

It is recommended that the now obsolete GPS90 system is replaced by a new and modern GPS system, i.e. the GPS08 system, in order to obtain even better performance of the INS. With a new system, the z coordinate could be utilized for height positioning as well as updating of the pitch state in the system model. This problem was evident in Test 6 (27, 28), Fig. 9.23, where the INS path became shorter due to the fact that the algorithms believed that the vehicle was moving at an upwards angle.

A new GPS system could also provide heading corrections in a reliable manner by filtering its data and only updating the yaw state in the system model when certain conditions are held, e.g. the vehicle must be on a straight path etc.

Further improvements could be made in the initial heading estimation. In its current form the NAV90 provides this information and it is not always of a high enough quality. If the initial heading is wrong then the system will follow the correct path but the path will be rotated by an amount of degrees that is equal to the initial heading error. This could be remedied by using the GPS information to update the yaw state in the system model, as previously mentioned.

For vehicles where it is critical for correct position and heading estimation, it is recommended to pursue the testing of the third gyroscope axis. This additional axis would, at least theoretically, provide a perfect reconstruction of the system and its dynamics.

A How to Run Simulations

Much about the behavior of CV90 can be explored by simulating the vehicle using either real or fictional data, the latter describing perhaps more extreme environmental scenarios. In the sequel it is assumed that the folder `CV90_Simulation_Suite` and all of its sub-folders are added to the MATLAB path. All simulations are run from the file `runSimulation.m`. If a run-time variable such as the duration of the simulation is changed, the simulation can be rerun by `run_simulation.m` which assumes a valid test case is already loaded. Other functions should in general not be altered[1] as this could have unforeseen consequences.

When a test case is set up and loaded it will be processed once and then saved to disk in order to minimize the time spent setting up a simulation. If any changes are found in the test case file, describing the test, this signal generation has to be performed again otherwise the saved test file on disk will be used. Sometimes errors can occur while saving or loading a test case why it is always a good idea to delete any old versions of the test case on disk if the program does not work as expected. For further information see Sec. A.6.

A.1 Real Test Cases

A.1.1 About NAV90 Logs

The foundation in a test case based on real sensor data is the actual sensor data, extracted from CV90. The log should optimally contain measurements in all the categories listed below, but in special cases it will be possible to run a simulation even if this is not the case. For example, UTC time must only be provided when comparing the simulation results to an external source such as a secondary GPS receiver. If PFP values are provided then the log file must also contain the UTC time for syncing purposes. If laser ranging was not performed during the real test run, and the only source for GPS measurements is GPS90, then only the first 11 inputs must be specified. The order in which they appear in the log does not matter.

```
'Tid'
'CAN-Buss:Vagnsfast vinkel elev'
'CAN-Buss:Vagnsfast vinkel sida'
'CAN-Buss:Vagnsfast vinkelhastighet elev'
'CAN-Buss:Vagnsfast vinkelhastighet sida'
'CAN-Buss:Gyrohastighet elev'
'CAN-Buss:Gyrohastighet sida'
'CAN-Buss:Vagnshastighet'
'Electronic Unit:Roll'
'Electronic Unit:Pitch'
```

[1] With one exception: the file `extract_data_from_bin.m` must be extended as more real test cases become available.

```
'Electronic Unit:Heading'
'Electronic Unit:UTC Tid'
'X7:Riktdon skytt PFP'
```

Given the above, a simulation using dead reckoning only can be carried out with the implemented Kalman filter turned off. An additional log file containing the GPS data from NAV90 must be supplied in order to fully take advantage of the devised algorithm. The log should optimally contain the following, and as before the order does not matter. While the UTC time is optional, some functionality will be lost should it not be supplied.

```
'Tid'
'Electronic Unit:Pos X'
'Electronic Unit:Pos Y'
'Electronic Unit:GPS Status'
'Electronic Unit:UTC Tid'
```

Specifying log files that contain other measurements than the ones mentioned above will cause the program to crash. It is assumed that the same logging equipment as already encountered will be used.

A.1.2 Define a Test Case

Given a standard NAV90 log, and optionally a NAV90 GPS log, a test case can be defined in MATLAB. It is assumed two files exist, conveniently named `nav90.log` and `gps.log`, respectively. The steps to define a test case are described below.

1. Type `edit extract_data_from_bin.m` in the MATLAB terminal. The file will contain previously performed tests, see Table 9.1, Table 9.2 and Table 9.3. Denote the new test by an integer case number that is not already in use. At the time of this publication numbers 1 through 28 were in use. If the test consist of two or more logs, a separate case must be set up for each log. Should there not exist a GPS log the variable `fileg` must not be defined. Two examples are shown below.

   ```
   case 99
       filel = 'nav90.log';
       fileg = 'gps.log';
   ```

   or alternatively, if no GPS data is available,

   ```
   case 99
       filel = 'nav90.log';
   ```

2. Create a new test case file by `File->New->Script` in MATLAB[2] and save it as `test_case_demo.m` for instance. The file must contain `test_case_` before the actual name or else the file will not be found. The file must contain the two lines

```
test_case = 99;
skovde_data_gen
```

   The first line specifies the case in `extract_data_from_bin.m` while the second line calls another script that handles the data extraction and the initialization of variables. Specifying a test that consist of two or more logs are as easy as naming all the case numbers, e.g.

```
test_case = [99, 100];
skovde_data_gen
```

3. The most basic type of test case is now defined. There exist a number of optional parameters, see Sec. A.3, that get initialized to default values. Depending on the situation, or how the test was performed, they can also be included in the test case.

A.1.3 An Example

The following is taken from the test case `amundtorp_1km.m`. The Kalman filter is on from $t = 0$ to $t = 40$, off from $t = 40$ to $t = 100$ and then on for the rest of the simulation[3]. All GPS coordinates are considered and validated by the tolerance `TOL`, set to 25 meters. The call to the function `extract_third_gyro_data.m` is included for consistency, even though no testing using a third gyro was carried out.

```
% OBLIGATORY
test_case = [19, 20];
skovde_data_gen

% OPTIONAL
% start angles (c = carrier, t = turret, s = sensor):
%   "c_alpha", "c_beta", "c_gamma", "t_alpha", "s_beta"
%       default: c_alpha = NAV90, c_beta = NAV90, c_gamma = 0,
%                t_alpha = SIDE_ANGLE(1), s_beta = ELEVATION_ANGLE(1)
c_alpha = [-2.5897,  -2.2480];
```

[2]The notation may differ in different versions of MATLAB but a blank `m`-file should be chosen.

[3]Note that the on/off switches are just a recommendation to the filter, e.g. if the vehicle is not moving then the filter is automatically turned off. Furthermore, while the times are absolute, the simulations will not always start at $t = 0$ but rather when the first valid GPS sample is encountered.

```
c_beta = [0, 0];

% simulink parameters:
%   "runTime"
%       on/off switches, empty/non-existent = ON entire simulation
%       default: Kalman filter is ON entire simulation
runTime = [0, 40, 1; 40, 100, 0; 100, -1, 1];

%   "measurements"
%       measurements, empty/non-existent = No available GPS data
%       default: all available GPS will be used


%   "TOL"
%       allowed difference for each x- and y-coordinate between GPS
%       and INS
%       default: 1e19 (i.e. all GPS coordinates are accepted)
TOL = 25;

% additional measurements (if available, no default values)
third_gyro = extract_third_gyro_data(fopen('strf90_24.txt'));
sec_gps_log = extract_sec_gps(load('COM8_121115_103553_cleaned.txt'));
target_id = 'TV-Masten 56501';
```

A.2 Fictional Test Cases

To create a fictional test case is slightly more complicated as all sensor signals first must be generated, the coordinates for the route must be calculated and all sensor errors must be specified. The necessary steps involved are described below.

A.2.1 Define a Test Case

1. Create a new test case file by `File->New->Script` in MATLAB and save it as `test_case_fictional_demo.m` for instance. The file must contain `test_case_` before the actual name or else the file will not be found. The file must contain at least the following variables or the program will crash.

   ```
   % simulation flags
   simul
   velocity

   % simulation sample times
   odometer_sample_rate
   ```

```
gyro_sample_rate
sample_rate
end_t

% carrier linear movement profiles
c_x_acc
c_y_acc
c_z_acc

% carrier angular movement profiles
c_alpha_acc
c_beta_acc
c_gamma_acc

% turret/weapon angular movement profiles
t_alpha_acc
s_beta_acc

init

% gyro specifications
M_0
lp_gain
tau_c
yaw_noise_gain
pitch_noise_gain
roll_noise_gain

% odometer specifications
refresh_rate
sigma_v
noise_gain

% GPS specifications
sample_real
gps_sample_rate
gps_refresh_rate
lim_r_r
offset
res
sigma
rand_gain
```

A more extensive treatment of each variable can be found below.

`'simul'`
: Boolean flag that indicates that this is a fictional test case. Must be set to `true`.

`'velocity'`
: Boolean flag that indicates if the specified movement profiles are accelerations or velocities. If `true` then it is assumed the profiles are velocities else they are accelerations. Often it is recommended to set this flag to `true`.

`'odometer_sample_rate'`
: Sample rate of the odometer sensor model. Should be set to 0.001.

`'gyro_sample_rate'`
: Sample rate of the gyro sensor model. Any value is allowed, but a too large value, e.g. 1, might result in poor performance of the sensor while a too small value, e.g. 0.001, will result in large amounts of data which might be too resource intensive. A recommended value is 0.01.

`'sample_rate'`
: Sample rate of the simulation. Should be set to the same value as `'gyro_sample_rate'`.

`'end_t'`
: Simulation end time, i.e. the duration of the simulation as the start time is automatically set to 0.

`'c_x_acc'`
: Linear velocity of CV90 along the x-axis, either specified directly or integrated once depending on the value of `'velocity'`. The variable should be a matrix with N rows and 2 columns, where the first column is time and the second column consist of the specified values. The number of rows, the time sample points, are determined by the sample rate of the odometer, i.e. `size(c_x_acc,1) == length(0:odometer_sample_rate:end_t)-1`.

`'c_y_acc'`
: Linear velocity of CV90 along the y-axis. Should be set to zero as the carrier only move along its own x-axis. For dimension arguments, see `'c_x_acc'`.

`'c_z_acc'`
: Linear velocity of CV90 along the z-axis. Should be set to zero as the carrier only move along its own x-axis. For dimension arguments, see `'c_x_acc'`.

`'c_alpha_acc'`
: Angular velocity of CV90 around the z-axis, either specified directly or integrated once depending on the value of `'velocity'`. The variable should be a matrix with N rows and 2 columns, where the first column is time and the second column consist of the specified values. The number of rows, the time sample points, are determined by the sample rate of the gyroscope, i.e. `size(c_alpha_acc,1) == length(0:gyro_sample_rate:end_t)-1`.

`'c_beta_acc'`
: Angular velocity of CV90 around the y-axis. For dimension arguments, see `'c_alpha_acc'`.

`'c_gamma_acc`
: Angular velocity of CV90 around the x-axis. Should usually be set to zero as the gyroscope cannot measure the velocity around this axis. For dimension arguments, see `'c_alpha_acc'`.

`'t_alpha_acc`
: Angular velocity of turret around the z-axis. For dimension arguments, see `'c_alpha_acc'`.

`'s_beta_acc`
: Angular velocity of weapon around the y-axis. For dimension arguments, see `'c_alpha_acc'`.

`'init'`
: Script that generates the specified signals. If the script is not called after the movement profiles are defined but before the sensor data is used the program will crash.

`'M_0'`
: Initial values for the integrator in the gyroscope sensor model for each axis. Given as a vector on the format `[`$\dot{\alpha}$`,` $\dot{\beta}$`,` $\dot{\gamma}$`]`.

`'tau_c'`
: Time constant affecting the bias stability error. Should be set to 600 s.

`'yaw_noise_gain'`
: Boolean flag vector that indicates which of the errors: bias stability, bias repeatability and random walk, respectively, should corrupt the measurement in the yaw axis.

`'pitch_noise_gain'`
: Boolean flag vector that indicates which of the errors: bias stability, bias repeatability and random walk, respectively, should corrupt the measurement in the pitch axis.

`'roll_noise_gain'`
: Boolean flag vector that indicates which of the errors: bias stability, bias repeatability and random walk, respectively, should corrupt the measurement in the roll axis.

`'refresh_rate'`
: Real sample rate of the odometer sensor model.

`'sigma_v'`
: Standard deviation for the odometer measurement.

`'noise_gain'`
: Boolean flag to indicate if the odometer signal is corrupted. If `true` then

the odometer readings are affected by an error with the standard deviation specified above, else not.

`'sample_real'`
: Sample rate of the simulation. Should be set to the same value as `'sample_rate'`.

`'gps_sample_rate'`
: How often the satellite is asked for new coordinates.

`'gps_refresh_rate'`
: Real sample rate of the GPS sensor model.

`'lim_r_r'`
: The variance in refresh rate. A recommended value is 0.05.

`'offset'`
: Constant error, i.e. the offset, affecting the GPS position given as a vector on the format `[x, y, z]`.

`'res'`
: Resolution of the GPS position for each axis. Given as a vector on the format `[x, y, z]`.

`'sigma'`
: Standard deviations for the GPS measurements given as a vector on the format `[`σ_x`, `σ_y`, `σ_z`]`.

`'rand_gain'`
: Boolean flag to indicate if the GPS signal is corrupted. If `true` then the GPS coordinates are affected of the errors specified above, else not.

2. The most basic type of test case is now defined. There exist a number of optional parameters, see Sec. A.3, that get initialized to default values. Depending on the situation they can also be included in the test case.

A.2.2 An Example

The following is taken from the test case `circle_and_turret_and_weapon.m`.

```
% OBLIGATORY
% simulation flags:
%   "simul"
%       must be set to 'true' for a fictional test case
%       default: false (i.e. MATLAB expects real data to be available)
simul = true;

%   "velocity"
%       if 'true' velocity profiles will be used to calculate the
%       velocity of the vehicle, if 'false' acceleration profiles
```

```
%       will be used
velocity = true;

% simulation parameters:
%   "odometer_sample_rate", "gyro_sample_rate", "sample_rate", "end_t"
%       the different sample rates for the sensor models and the
%       simulation end time (the start time is assumed to be t = 0 s)
odometer_sample_rate = 0.001;
gyro_sample_rate = 0.01;
sample_rate = gyro_sample_rate;
end_t = 600;

% carrier movement profiles:
%   specify the acceleration of the carrier (velocity == false) or the
%   velocity of the carrier (velocity == true)
%   for each sample there must be an associated velocity/acceleration

%   linear movement (acceleration or velocity):
%       "c_x_acc", "c_y_acc", "c_z_acc"
CLA = length(0:odometer_sample_rate:end_t-odometer_sample_rate);
temp = [(70/3.6)*ones(1, CLA), zeros(1, 0)];
c_x_acc = [0:odometer_sample_rate:end_t-odometer_sample_rate; temp]';
c_y_acc = [0:odometer_sample_rate:end_t-odometer_sample_rate; ...
    zeros(1, CLA)]';
c_z_acc = [0:odometer_sample_rate:end_t-odometer_sample_rate; ...
    zeros(1, CLA)]';

%   angular movement (acceleration or velocity):
%       "c_alpha_acc", "c_beta_acc", "c_gamma_acc"
CAA = length(0:sample_rate:end_t-sample_rate);
N = CAA;

temp = [(pi/75)*ones(1, N), zeros(1, CAA-N)];
c_alpha_acc = [0:sample_rate:end_t-sample_rate; temp]';

temp = [0*sin(1/100 * (0:sample_rate:end_t-sample_rate)), ...
    zeros(1, CAA-N)];
c_beta_acc  = [0:sample_rate:end_t-sample_rate; temp]';

temp = [zeros(1,CAA), zeros(1, CAA-N)];
c_gamma_acc = [0:sample_rate:end_t-sample_rate; temp]';

% gyro movement profiles:
```

```
%   specify the acceleration of the gyro (velocity == false) or the
%   velocity of the gyro (velocity == true)
%   for each sample there must be an associated velocity/acceleration

%   turret angular movement:
%       "t_alpha_acc"
temp = [2*pi/10*cos(1/10*(0:sample_rate:end_t-sample_rate)), ...
    zeros(1, CAA-N)];
t_alpha_acc = [0:gyro_sample_rate:end_t-gyro_sample_rate; temp]';

%   sensor angular movement
%       "s_beta_acc"
temp = [0.1/10*cos(1/10*(0:sample_rate:end_t-sample_rate)), ...
    zeros(1, CAA-N)];
s_beta_acc = [0:gyro_sample_rate:end_t-gyro_sample_rate; temp]';

% generate signals, this script MUST be called before the sensor
% models are set up and used
init

% sensor model parameters:
%   set up gyro errors
%   set up the odometer and interpolate using ZOH
%   set up GPS

%   gyro:
%       "M_0", "lp_gain", "tau_c", "yaw_noise_gain",
%       "pitch_noise_gain", "roll_noise_gain"
%
% the initial values for the gyro sensor model in each axis
M_0 = [t_alpha_vel_real(1); s_beta_vel_real(1); 0];

% if the low-pass filter should be turned on, 1, or off, 0, for the
% different gyro axes (yaw, pitch, roll)
lp_gain = [1, 1, 1];

% time constant
tau_c = 600;

% if the signal should be corrupted, rand_gain = 1, or not,
% rand_gain = 0 for each of the three errors (bias stability,
% bias repeatability, random walk) affecting the different
% gyro axes (yaw, pitch, roll)
```

```
yaw_noise_gain = [1, 1, 1];
pitch_noise_gain = [1, 1, 1];
roll_noise_gain = [1, 1, 1];

%   odo:
%       "refresh_rate", "sigma_v", "noise_gain"
%
% real sample rate for the odometer sensor model.
% if less than one, linear interpolating will be used
refresh_rate = 1;

% standard deviation for each measurement
sigma_v = 0.01;

% if the signal should be corrupted, rand_gain = 1, or not,
% rand_gain = 0
noise_gain = 1;

%   GPS:
%       "sample_real", "gps_sample_rate", "gps_refresh_rate",
%       "lim_r_r", "offset", "res", "sigma", rand_gain"
%
% time vector used in simulink
sample_real = sample_rate;

% how often the satellite is asked for new coordinates
gps_sample_rate = 0.2;

% the real sample rate for the gps
gps_refresh_rate = 1;

% how much the refresh rate varies during the simulation
lim_r_r = 0.05;

% constant error, i.e. the offset on the gps-signal (x,y,z)
offset = [1, 1, 1];

% the resolution of the gps (x,y,z)
res = [0.1, 0.1, 0.1];

% standard deviation for each measurement
sigma = [0.1, 0.1, 0.1];
```

```
% if the signal is corrupted, rand_gain = 1, or not, rand_gain = 0
rand_gain = 1;


% OPTIONAL
%   "name"
%       the title of the simulation
%       default: 'Untitled'
name = 'Circle with turret/weapon movement';

%   "type_of_gyro"
%       if '2' a two-axis gyro will be used in the simulation,
%       if not '2' a three-axis gyro will be used
%       default: 2
type_of_gyro = 2;

%   "start"
%       start position of carrier (x,y,z)
%       default: [0, 0, 0]
start = [0, 0, 0];

% start angles (c = carrier, t = turret, s = sensor):
%   "c_alpha", "c_beta", "c_gamma", "t_alpha", "s_beta"
%       default: c_alpha = 0, c_beta = 0, c_gamma = 0
%                t_alpha = SIDE_ANGLE(1), s_beta = ELEVATION_ANGLE(1)
c_alpha = pi/4;
c_beta = 0;
t_alpha = -pi/4;
s_beta = pi/180;

% simulink parameters:
%   "runTime"
%       on/off switches, empty/non-existent = ON entire simulation
%       default: Kalman filter is ON entire simulation
%       (runTime = [1,-1,1])
runTime = [0, 100, 1; 100,  -1, 0];

%   "measurements"
%       measurements, empty/non-existent = No available GPS data
%       default: all available GPS will be used
```

A.3 Optional Run-time Parameters

In addition to the variables that must be defined in order for the program to run there exist a number of other variables that can be used to control the simulation, such as initial values for the carrier angles, available measurements and so forth.

`'name'`
: Describing title for the current simulation.
Default: NAV90 log ID. If no valid log exist the default value is `'Untitled'`.

`'type_of_gyro'`
: The number of axes for the gyroscope in use. Variable is only considered if `simul == true`.
Default: 2.

`'start'`
: Start position for the carrier in the ground frame.
Default: If GPS data is available the start position is specified by the first GPS measurement, else `[0, 0, 0]`.

`'c_alpha'`
: Initial yaw angle of the carrier. The size should be at most the size of `test_case`. If only one value is provided while the the test is based on two NAV90 logs the program will prompt for a new initial heading when the simulation restarts after the log switch. If no new heading is provided the latest one computed will be used.
Default: NAV90 heading at $t = 0$ or, if GPS available, at the time of the first GPS sample.

`'c_beta'`
: Initial pitch angle of the carrier. The size should be at most the size of `test_case`. If only one value is provided while the the test is based on two NAV90 logs the simulation will restart using the latest angle computed.
Default: NAV90 pitch at $t = 0$ or, if GPS available, at the time of the first GPS sample.

`'c_gamma'`
: Initial roll angle of the carrier. The size should be at most the size of `test_case`. If only one value is provided while the the test is based on two NAV90 logs the simulation will restart using the latest angle computed.
Default: 0.

`'t_alpha'`
: Initial turret angle, used only in an animation of the carrier movement.
Default: Side angle at $t = 0$.

`'s_beta'`
: Initial weapon angle, used only in an animation of the carrier movement.
Default: Elevation angle at $t = 0$.

`'runTime'`
: Kalman filter switches indicating if the filter should be either on, `true`, or off, `false`. The format is `[from_time, to_time, on_off]`. Use -1 to specify the simulation end time. Once -1 is encountered in the matrix, the following switches will be ignored.
Default: `[1, -1, 1]`.

`'measurements'`
: Matrix containing the available GPS data, if any. By removing values in the beginning, middle or the end of `measurements`, scenarios such as a lost GPS connection could be simulated. While possible, it is not, however, recommended to remove values from this matrix direcly but rather use the Kalman filter switches to turn the filter on/off. No GPS will be available if `measurements = []`, with the result that the Kalman filter is turned off for the duration of the simulation. The format is `[gps_time, gps_x, gps_y, gps_z, gps_heading, gps_status]`.
Default: The matrix contains all available data.

`'TOL'`
: Tolerance to determine which GPS coordinates are trustworthy for use in the Kalman filter. As the vehicle is assumed to follow the laws of physics, a GPS position too far from the last know position is not feasible. The condition that must be fulfilled for a GPS measurement to be valid is `abs(x_GPS - x_INS) < TOL && abs(y_GPS - y_INS) < TOL`.
Default: 10^{19}.

`'coordinate_system'`
: String indicating how the positive z-axis should be defined, either `'up'` or `'down'`. The NAV90 logs are not consistent in how they define the coordinate system why this variable may be used in some cases.
Default: `'down'`.

`'sec_gps_log'`
: Measurements from a secondary GPS receiver used to evaluate the performance of NAV90. The log should be given on the format `[gps_utc_time, gps_x, gps_y]`.
Default: Undefined variable.

`'target_id'`
: The display name for a given laser ranging target. The name must be a string found in description output from the function `extract_waypoints.m`.
Default: `''`.

`'do_animation'`
: Boolean flag that indicates if generated sensor data also should be animated. The function that calculates uncorrupted coordinates for a fictional test case also has the functionality to animate the result, which might be unnecessary for a simple simulation, why the flag can be used to disable this feature. If `true` the input data

is animated, else not.
Default: `false`.

`'axes'`
Vector containing axis properties on the format `[x_min, x_max, y_min, y_max, z_min, z_max]`. The variable act as an additional input parameter to the function `animate_simulation.m`. If not specified the axes will resize automatically which might be undesirable.
Default: Undefined variable, i.e. automatic scaling.

A.4 Plot Results

There are a couple of different plot options available. Each are described in more detail below.

`plot_coordinates.m`
Plots the route specified in the test case file calculated by dead reckoning. If the Kalman filter is on during the simulation the function also plots any valid GPS measurements from NAV90 as well as markers that indicate which coordinates were used by the filter in the computations. If there is secondary GPS data available, this route is also added to the plot. If the test case is composed of two or more NAV90 logs it is indicated in the plot at which coordinates the log switch occurred. The function will try to plot target statistics from laser ranging but this will only succeed if the given NAV90 logs contain PFP values.

`plot_position_absolute_error.m`
Plots the absolute error, $\epsilon = \sqrt{(x_{GPS} - x_{INS})^2 + (y_{GPS} - y_{INS})^2}$, for each GPS coordinate and the equivalent INS coordinate. Any GPS position that is deemed valid by the specified `TOL` will be used in the comparison. If no GPS data is available the plot will be empty.

`plot_position_north_east_error.m`
Plots the northing error, $\epsilon_y = y_{GPS} - y_{INS}$, as well as the easting error, $\epsilon_x = x_{GPS} - x_{INS}$, for each GPS coordinate and the equivalent INS coordinate. Any GPS position that is deemed valid by the specified `TOL` will be used in the comparison. If no GPS data is available the plot will be empty.

There also exist the following two functions `plot_heading_ins_vs_ublox_GPS` and `plot_position_error_ins_vs_ublox_GPS`, in addition to the plot options mentioned above, which compare the INS heading and position, respectively, to measurements from an external GPS. These functions will only work if such measurements are specified in the test case on the format

```
sec_gps_log = extract_sec_gps(load('COM8_121115_103553_cleaned.txt'));
```

where the loaded file name should refer to a GPS log file. The functionality is highly dependant on the log file format and thus the functions will not plot anything if the formatting is wrong, or if the log file does not contain the necessary information. As the main goal with this thesis was to compare INS performance to NAV90, and GPS90, no further comments about the functions are included.

A.5 Simulation Output

The simulation model produces a number of outputs where some, e.g. `time_ind`, are only intended for use by private functions. A rundown is presented below.

`'coord_lin_ss'`
: Matrix containing the simulated x, y and z coordinates. The first column contain the x coordinates at times specified by `simulation_time`, the second column contain the y coordinates while the third column contain the z coordinates.

`'angles_lin_ss'`
: Matrix containing the simulated yaw and pitch angles. The first column contain the carrier headings at times specified by `simulation_time` while the second column contain the carrier pitch angles.

`'real_coord'`
: GPS coordinates from `measurements` that are deemed valid by `TOL`. Only x and y coordinates are considered in the current implementation.

`'kalmanOn'`
: Matrix containing the boolean flags `[isMoving, change, gpsValid, onOff]` used by the Kalman filter. Is processed to determine at which time instances the filter was turned on.

`'last_states'`
: Matrix containing the simulated last four states of the full state space model at times specified by `simulation_time`.

`'time_ind'`
: Matrix containing the current count, i.e. what GPS measurement is currently in use, and the current time. The format is `[time, count, time, count]`. Columns three and four are $\neq 0$ as soon as a valid GPS sample is available while columns one and two only are $\neq 0$ if a filter correction have taken place. Functionality overlap with `kalmanOn`.

`'simulation_time'`
: Time vector used in the simulation.

`'simulation_count'`
: Vector containing the count for which measurement is currently in use by the Kalman filter. Is used to restart simulations.

A.6 Possible Sources of Errors

There are always ways to misuse a program, intentionally or otherwise. Below is a list of possible errors and what may have caused them. If the simulation suite is used in the manner described earlier, there should not be a problem.

MATLAB crashes
This usually occurs if the some of the SIMULINK models have been compiled on a machine with a different computer architecture, e.g. the files are compiled on a 64-bit machine and then run on a 32-bit machine. Removing all old compilation files in the folder `Simulation_Models` should solve the problem.

A function cannot be found
If the first simulation during a session is not initiated from `runSimulation.m` the path of the simulation suite might not have been added to the general MATLAB path which will result in a an error saying that any chosen function cannot be found. The solution is to always start a fresh simulation session using the `runSimulation.m` function, which automatically will take care of the path problem. It should be noted, however, that the path obviously can be set manually.

A test case cannot be loaded
This probably is a result from the formatting of the specified log files or if the files cannot be found, see above. The most comman cause is if the given NAV90 log contains too much, or too little, information as this will cause the extract function to fail. Check the log file so that the format is correct. If this does not solve the problem, try to remove any old `mat`-files on disk describing the chosen test case.

The plot functions do not plot additional GPS measurements
If the plot functions are used on a test case with no measurements from an external GPS they will not work. Furthermore, it is not possible to compare the INS heading and/or position to a different GPS than GPS90 if the given NAV90 log does not contain a column with the UTC time.

The laser ranging plot does not work
A bug was found with the NAV90 log file ripper program with the result that NAV90 logs do not always contain information about PFP, which is used for laser ranging. If laser ranging did take place during a test but does not show up in the plot window after a simulation this bug might be the cause. Another cause could be that the given target id string does not exist in the current implementation of the program.

B Animate Raw Sensor Data

VISUALIZATION IS A BIG PART of understanding how a complex system functions, which is why much work was dedicated to create a good graphical representation of available measurements. Using either real sensor data from CV90 or generated sensor data, the nonlinear system model is used to calculate, based on a start position, how the vehicle behaves. The position of the carrier, in the ground frame, is animatated in an iterative process as a series of three-dimensional plots by the script `animate_simulation.m`.

If a proper test case is loaded and previously have been animated, the alternative function `play_movie.m` may be used which does not redo the calculations.

B.1 I/O

The input/output of the function `plot_simulation.m` are described below. All input variables except `do_animation`, which should be set to `true`, and `axes` are already defined[4] if a test case is used that follows the previously established guidelines. The function call is

```
[h, angles, coord, M] = plot_simulation(carrier_tra_vel, ...
     carrier_ang_vel, s_beta_vel_real, t_alpha_vel_real, t, ...
     sample_rate, do_animation, c_alpha, c_beta, t_alpha ...
     s_beta, start, coordinate_system, axes);
```

The different input parameters are summerized below, for further explanation see Sec. A.2.1 and Sec. A.3.

`'carrier_tra_vel'`
: Linear velocity of the carrier.

`'carrier_ang_vel'`
: Angular velocity of the carrier.

`'s_beta_vel_real'`
: Speed of the weapon elevation.

`'t_alpha_vel_real'`
: Speed of the turret rotation.

`'t'`
: Simulation time.

`'sample_rate'`
: Time step.

[4] Some of the variables, e.g. `t`, are implicitly defined based on either the specified NAV90 log or the simulation end time.

`'do_animation'`
: Boolean flag.

`'c_alpha'`
: Initial carrier heading.

`'c_beta'`
: Initial carrier pitch angle.

`'t_alpha'`
: Initial turret rotation.

`'s_beta'`
: Initial weapon elevation.

`'start'`
: Start position.

`'coordinate_system'`
: Direction of positive z-axis.

`'axes'`
: Plot window axes.

The function output is

`'h'`
: Figure handle to plot window used in the animation.

`'angles'`
: Matrix containing the calculated yaw, pitch, roll, turret rotation and weapon angles at times specified by `t`. The first column contain the carrier headings, the second column the carrier pitch angles, the third column the carrier roll angles, the fourth column the turret rotation angles while the fifth column contain the weapon elevation angles.

`'coord'`
: Matrix containing the calculated x, y and z coordinates at times specified by `t`. The first column contain the x coordinates, the second column contain the y coordinates while the third column contain the z coordinates.

`'M'`
: MATLAB cell array structure containing all the frames from the animation. Used by the function `play_movie.m`.

References

[1] Petteri Aimonen. *Basic Concept of Kalman Filtering*. Accessed 3-December-2012. 2012. URL: www.en.wikipedia.org/wiki/File:Basic_concept_of_Kalman_filtering.svg.

[2] BAE Systems. *GPS/INS för Banstyrning*. 02794855/01. 2001.

[3] Per-Erik Bergman. *Rotation and Translation in 2D Euclidian Space*. Accessed 3-December-2012. 2010. URL: www.jayway.com/wordpress/wp-content/uploads/2009/12/RotateTranslate.png.

[4] Per-Erik Bergman. *Translation and Rotation in 2D Euclidian Space*. Accessed 3-December-2012. 2010. URL: www.jayway.com/wordpress/wp-content/uploads/2009/12/TranslateRotate.png.

[5] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Second Edition. Prentice Hall, 1989. ISBN: 0201095289.

[6] Tore Hägglund. *Reglerteknik AK, Föreläsningar*. Spring Semester 2008. Institutionen för Reglerteknik, Lunds Tekniska Högskola, 2008.

[7] Rolf Johansson. *Predictive and Adaptive Control*. Fall Semester 2010. Department of Automatic Control, Lund University, 2010.

[8] Försvarets Materialverk. *Stridsfordon 90, Historik*. Accessed 27-November-2012. 2012. URL: www.fmv.se/sv/Projekt/Stridsfordon-90/Historik/.

[9] D. McFadden. *Diagram of a Fiber Optic Gyroscope*. Accessed 3-December-2012. 2012. URL: www.en.wikipedia.org/wiki/File:Fibre-optic-interferometer.svg.

[10] Christer Nyberg. *Mekanik, Fortsättningskurs*. First Edition. Liber, 2006. ISBN: 9147084022.

[11] Erik Nygård. *Comparison between GPS90, NAV90 and an INS prototype based on Gyroscope Data*. BAE Systems AB. 2012.

[12] Erik Nygård. *Overview of the NAV90 System*. BAE Systems AB. 2012.

[13] Lars Pihlström. *CV90 with 3D Camouflage, Front View*. BAE Systems AB. 2012.

[14] Lars Pihlström. *CV90 with 3D Camouflage, Front-side View*. BAE Systems AB. 2012.

[15] Saab Avionics AB. *Specification, Fiber Optic Gyro (F.O.G.), Part No: 8088 000-102*. R-8088902-102. 2001.

[16] Anja Schönhardt. *An Overview of the Geometry of Yaw, Pitch and Roll Angles*. Accessed 3-December-2012. 2012. URL: www.doas-bremen.de/images/idoas/idoas_angles.png.

[17] Jorge Stolfi. *Cartesian Coordinate System*. Accessed 3-December-2012. 2012. URL: www.en.wikipedia.org/wiki/File:Coord_system_CA_0.svg.

[18] The Analytic Sciences Corporation The Technical Staff. *Applied Optimal Estimation*. First Edition. The M.I.T. Press, 1974. ISBN: 0262570483.

[19] D. H. Titterton and J. L. Weston. *Strapdown Inertial Navigation Technology*. First Edition. Peter Peregrinus Ltd. on behalf of the Institution of Electrical Engineers, 1997. ISBN: 1563476932.

[20] Wikipedia. *Combat Vehicle 90*. Accessed 27-November-2012. 2012. URL: www.en.wikipedia.org/wiki/Combat_Vehicle_90.

[21] Karl-Erik Årzen. *Real-Time Control Systems*. Department of Automatic Control, Lund University, 2009.