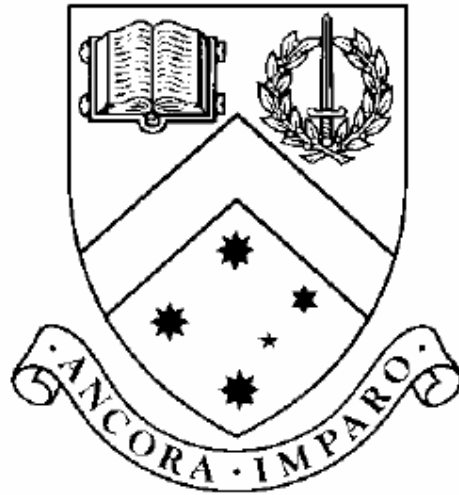


# Unmanned Aerial Vehicles

## Development of Navigation and Mission Planning

By

Ognjen Jovanovic



Thesis

Submitted by Ognjen Jovanovic

in partial fulfilments of the Requirements for the Degree of

Bachelor of Digital Systems with Honours (1200)

Supervisors: Dr Gordon Lowe

Assoc. Prof. Bijan Shirinzadeh

**Clayton School of Information Technology**

**Monash University**

November 2006

© Copyright

by

Ognjen Jovanovic

2006

To my parents.

# Contents

<b>LIST OF FIGURES</b> .....	<b>VI</b>
<b>LIST OF TABLES</b> .....	<b>VII</b>
<b>ABSTRACT</b> .....	<b>VIII</b>
<b>ACKNOWLEDGMENTS</b> .....	<b>X</b>
<b>1.0 INTRODUCTION</b> .....	<b>1</b>
<b>1.1 Applications of Unmanned Aerial Vehicles</b> .....	<b>1</b>
<b>1.2 Helicopter Flight</b> .....	<b>2</b>
<b>1.4 UAV Design</b> .....	<b>3</b>
<b>1.5 Thesis Overview</b> .....	<b>4</b>
<b>2.0 MISSION PLANNING</b> .....	<b>5</b>
<b>2.1 Requirements</b> .....	<b>5</b>
<b>2.2 Mission Controller</b> .....	<b>6</b>
<b>2.3 Communication</b> .....	<b>7</b>
<b>2.4 Mavionics</b> .....	<b>7</b>
<b>3.0 PATH PLANNING</b> .....	<b>9</b>
<b>3.1 Path Planning Approaches</b> .....	<b>9</b>
<b>3.2 Extended Local Optimization Strategy (ELOS)</b> .....	<b>10</b>
<b>4.0 NAVIGATION</b> .....	<b>12</b>
<b>4.1 Background</b> .....	<b>12</b>
<b>4.2 Inertial Navigation</b> .....	<b>13</b>
<b>4.2 Reference Frames and Transformations</b> .....	<b>15</b>
<b>4.3 Inertial Navigation Equations</b> .....	<b>19</b>
4.3.1 Position and Velocity Equations.....	19
4.3.2 Attitude Equations.....	20
<b>4.4 Inertial Navigation Error Sources</b> .....	<b>21</b>
<b>4.5 GPS</b> .....	<b>22</b>
4.5.1 INS/GPS Integration.....	23

<b>4.6 Kalman Filter</b> .....	<b>24</b>
4.6.1 Kalman Filter Equations.....	25
<b>4.7 Summary</b> .....	<b>27</b>
<b>5.0 IMPLEMENTATION</b> .....	<b>28</b>
<b>5.1 OHBY Mission Planning System</b> .....	<b>28</b>
5.1.1 Graphical User Interface.....	33
<b>5.2 ELOS Path Planning</b> .....	<b>36</b>
5.2.1 ELOS 2D Path Planning.....	37
5.2.2 ELOS 3D Path Planning.....	38
5.2.3 Threats and No Fly Zones.....	40
<b>5.3 INS module</b> .....	<b>41</b>
5.3.1 Inertial Measurement Unit.....	41
5.3.1 Attitude Integration.....	42
5.3.2 Velocity and Position Integration.....	44
5.3.3 Summary.....	45
<b>5.4 Kalman Filter Module</b> .....	<b>45</b>
<b>5.4.1 Error Dynamics</b> .....	<b>46</b>
<b>6.0 RESULTS &amp; DISCUSSION</b> .....	<b>50</b>
<b>6.1 OHBY Mission Planning</b> .....	<b>50</b>
<b>6.2 ELOS Path Planning</b> .....	<b>50</b>
6.2.1 ELOS 2D Algorithm.....	51
6.2.2 ELOS 3D Algorithm.....	54
6.2.3 No-Fly Zones Extension.....	55
<b>6.3 Navigation</b> .....	<b>57</b>
6.3.1 IMU.....	58
6.3.2 INS.....	58
6.3.3 Kalman Filter.....	59
<b>7.0 CONCLUSION &amp; FUTURE WORK</b> .....	<b>61</b>
7.1 Limitations.....	61
7.2 Future Work.....	62
<b>8.0 REFERENCES</b> .....	<b>63</b>
<b>APPENDIX A</b> .....	<b>66</b>
<b>APPENDIX B</b> .....	<b>68</b>
<b>APPENDIX C</b> .....	<b>70</b>

# List of Figures

Figure 1: Mavionics screenshot.....	8
Figure 2: Inertial frame and earth frame coordinates comparison.....	16
Figure 3: Navigation frame coordinates from.....	17
Figure 4: Body frame coordinates, from.....	17
Figure 5: INS.....	21
Figure 6: Feed-forward aided INS.....	24
Figure 7: Feed-back aided INS.....	24
Figure 8: The Kalman Filter loop.....	27
Figure 9: Flight envelope translation.....	29
Figure 10: Mission controller interaction diagram.....	32
Figure 11: OHBY Main screen.....	33
Figure 12: Mission List display.....	35
Figure 13: Edit Mission menu.....	36
Figure 14: 3D neighbourhood extraction.....	39
Figure 15: Block Diagram of INS mechanisation.....	45
Figure 16: ELOS 2D Path Planning: Test 1.....	51
Figure 17: ELOS 2D Path Planning, Test 2.....	52
Figure 18: Time snapshots of the 2D array.....	53
Figure 19: 3D path planner.....	54
Figure 20: No-fly Zone path generation, Side view.....	56
Figure 21: No-fly Zone path generation, Top view.....	57
Figure 22: INS Latitude and Longitude.....	59
Figure 23: Kalman Filter results.....	60

## List of Tables

Table 1: INS errors, adapted from Kumar (2004).....	22
Table 2: Way-point structure.....	30
Table 3: Mission commands.....	31
Table 4: Interactive object and way-point input.....	34

# Unmanned Aerial Vehicles

## Development of Navigation and Mission Planning

Ognjen Jovanovic

[Ognjen.Jovanovic@csse.monash.edu.au](mailto:Ognjen.Jovanovic@csse.monash.edu.au)

Monash University, 2006

Supervisor: Dr Gordon Lowe

Assoc. Prof. Bijan Shirinzadeh

### Abstract

Unmanned Aerial Vehicle (UAV) is increasingly popular to private and civil sectors, due to their versatility and ability to perform a wide variety of applications. This research extends upon research in progress at Monash University, which aims to develop an UAV, specifically the rotary UAV, at minimal cost.

In this thesis, several crucial components of an UAV system have been designed and implemented. Namely, these components are the mission and path planning, in addition to the navigation system. The OHBY mission planner was developed as a framework that is essential for a ground station to operate a UAV. Currently, OHBY enables mission planning via user specified way-points. Furthermore, a path planning algorithm enables the UAV to avoid obstacles.

The components were tested via simulations, which ultimately confirmed the validity of their functionality. Unfortunately, the implementation of the navigation system was not as successful. Nevertheless, test results suggest that the INS is operational; however, the integration of GPS via the Kalman filter was not as effective. These limitations could not be rectified due to time constraints. Nevertheless, the components in their current state provide a great deal of functionality and a beneficial framework for potential future work.

# Unmanned Aerial Vehicles

## Development of Navigation and Mission Planning

### Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

---

Ognjen Jovanovic  
November 7, 2006.

# Acknowledgments

I would like to thank the following people:

- My family for all their love and support.
- The honours students for bearing with me.
  - Bart for all his help and sometimes motivation.
  - Yasir for proof reading my thesis.
  - Huddia for being the mum of the honours lab and for proof reading my thesis.
- My Supervisors Dr. Gordon Lowe and Assoc. Prof. Bijan Shirinzadeh for their guidance.
- Charles Grief and Tim Ferguson for their enthusiasm and interest during the year.

Ognjen Jovanovic

*Monash University*  
*November 2006*



## ***1.0 Introduction***

The development of an Unmanned Aerial Vehicle (UAV) is an ongoing project conducted by the Monash University's Robotic and Mechatronic Research Laboratory (MURMRL). The project began in 2001 with a goal to develop a low cost rotary UAV that could be deployed in real life situations to perform meaningful tasks.

An UAV is an aircraft capable of pilot-less flight; it must achieve such flight by generating autonomous control movements. Furthermore, a UAV must be capable of satisfying mission requirements. The American Department of Defence Dictionary defines an UAV as:

A powered aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expandable or recoverable, and can carry a lethal or non-lethal payload. Ballistic or semi-ballistic vehicles, cruise missiles, and artillery projectiles are not considered unmanned aerial vehicles.

### ***1.1 Applications of Unmanned Aerial Vehicles***

Originally, the military was the main driving force behind the development of UAVs. However, in recent times a considerable amount of effort and technology has been directed into UAVs for use in private and civil sectors. Currently, an overwhelming majority of UAVs utilised around the world are Japanese produced helicopters used for agricultural purposes; performing approximately one third of the total agricultural aviation in Japan (Newcome, 2004).

UAVs provide significant advantages in comparison to traditional manned aircraft. For example, in dangerous missions they can eliminate or minimize the risk to human life. UAVs could also reduce operational cost while improving work efficiency. With the improvement in technology, many mundane and repetitive everyday tasks have been automated. Thus, UAVs in some instances have been developed to perform such repetitive tasks. Deployment of UAVs for repetitive tasks generally results in an improvement in work efficiency. This is due to the high repeatability and accuracy of UAVs (Saggiani & Teodorani, 2003).

There are possible UAV applications in various fields of both the military and civil sectors. Some of the possible deployments of UAVs include:

- Evaluation and monitoring of environmental conditions
- Search and rescue support operations
- Security, surveillance and reconnaissance
- Agriculture, monitoring fence lines and crop spraying
- Terrain mapping

UAVs are generally fixed wing or rotary. A rotary UAV is a helicopter type UAV. This thesis is concerned with the development of a rotary UAV; due to the unique flight capabilities of the aircraft.

## ***1.2 Helicopter Flight***

The modern helicopter is a complex and versatile machine. Its complexity and versatility is due to the unstable nature of the helicopter. Unlike other aircraft, the helicopter is able to move freely in a three dimensional space. Its main attraction is its manoeuvrability. The helicopter, besides the standard flight manoeuvres is also capable of:

- vertical take-offs and landings
- maintaining motionless hover
- flying backwards
- rotating on the spot (pirouette)

The trade-offs for the increased manoeuvrability, besides system complexity, are the reduced flight speed, and thus increased power consumption. This in turn limits load and flight capacity of helicopters (Watkinson, 2004). However, the omni-directional capabilities of the helicopter make it highly suitable for many of the applications listed above. This forms the basis of the interest in the development of a low cost rotary UAV.

## **1.4 UAV Design**

An autonomous UAV system consists of complex hardware and software components. These components can be categorized into three main research areas as proposed by Quach et al. (2004), in their research at Monash University. These areas are:

- Sensor system
- Control system
- Navigation and Guidance systems

These components are yet to be implemented. Furthermore, this proposed system is fairly broad and completely ignores the need for a mission planning component. Mission planning capabilities are essential for a UAV system so that it can perform meaningful tasks. The development of a mission planning system would be the first step in developing a UAV ground station, which would give an operator complete control of the UAV. Furthermore, navigation and guidance systems are essential for the UAV to execute planned missions.

Navigation and guidance systems require a sensor system to be implemented. The navigation should use inertial data from the sensor system, to perform inertial navigation. The inertial navigation output must be integrated with GPS to achieve adequate performance. The sensor fusion of GPS and inertial navigation is performed using a Kalman filter. Furthermore, the guidance component will require a method to guide the UAV to a target location

through an optimal collision free path. This is known as path planning, and thus a path planning algorithm must also be implemented.

The focus of this thesis is the development of a sensor and navigation system, as well as the mission and path planning. The design and implementation of the control algorithm is the focus of research of the Engineering students involved with this project. Therefore, the design, development and implementation results of the navigation, mission and path planning systems are only discussed.

### ***1.5 Thesis Overview***

Chapter 1 has introduced the concept of Unmanned Aerial Vehicles and the motivation for this research. Chapter 2 discusses Mission Planning; a core component of a UAV system. Path planning for a UAV is examined in Chapter 3. Navigation for a UAV is presented in Chapter 4; specifically Inertial Navigation and its integration with GPS using a Kalman filter. Chapter 5 is concerned with the implementation of the different components and the results are discussed in Chapter 6. Finally, Chapter 7 concludes the thesis and proposes future work to be performed to further the development of the Monash UAV project.

## ***2.0 Mission Planning***

This chapter discusses design requirements of a Mission Planning System. Ground station software as well as the onboard software is examined, with a brief overview of the communication requirements between the two entities.

### ***2.1 Requirements***

An UAV would be a scarce asset to its owner, and maximising its efficiency and effectiveness would be of utmost importance. Therefore a Mission Planning System (MPS) is required to plan, manage and execute missions effectively. The goal of the MPS is not to create missions rather it is to assist the operator in the development of UAV missions.

Defence Science and Technology Organisation (DSTO), in 1995, performed a survey on MPS requirements. The report determined baseline and desirable features of a MPS. The primary objective of a MPS, according to the report, is to insure planned missions satisfy information/mission requirements, whilst incorporating any limitations imposed. Limitations to consider may include:

- UAV payload
- Flight time
- UAV dynamic constraints
- Weather
- Any fuel limitations

The DSTO (1995) report identified the importance of organised databases, which should store information such as way-points, maps of operation and known threats. Furthermore, it was deemed essential that these databases are easily editable. In addition, a decision and analyses component, which could assist decision making and mission analyses, was identified as essential in a baseline MPS. Finally, it was determined that an easy to operate graphical user interface is essential.

Hall (1988) believes a good user interface is crucial for the wider acceptance of a technology. Therefore, a well designed user interface is of utmost importance. However, a major challenge of developing a user interface is the graphical representation of the mission; specifically, the amount of mission information to display. The user interface needs to present as much information as is necessary, without confusing the user (Hall, 1988). This can be a difficult design hurdle; therefore, allowing the user full control of what is displayed could overcome this difficulty in design (DSTO, 1995). Furthermore, an interface with usability in mind should have a simple process for data input.

## ***2.2 Mission Controller***

The mission controller is located on-board the UAV. It is required to execute actions based on sensor data and mission plan. These actions may be to control the transmitter (relay information back to ground station) or the rotors to perform a manoeuvre. The autonomy of the mission controller will greatly determine the complexity of the mission plan required (DSTO, 1995). A simple pre-programmed controller which simply travels from way-point to way-point requires a highly detailed mission plan. In contrast, a more capable mission controller will require a far simpler mission plan.

Ideally, an autonomous mission controller would be goal-directed and capable of adapting to changes in the operating environment. However, this level of autonomy is currently unavailable. Therefore, the mission controller to be implemented may not be completely autonomous; however, it could be more advanced than a simple pre-programmed controller. The major requirement of the mission controller will be to safely guide the UAV to its target location, whilst avoiding any obstacles. To achieve this, the mission controller must have a path planning sub-component and some method of object detection. A suggested approach for object detection is via an omni-

directional video camera, this is out of scope of this thesis due to time constraints. However, path planning is discussed in the next Chapter.

### ***2.3 Communication***

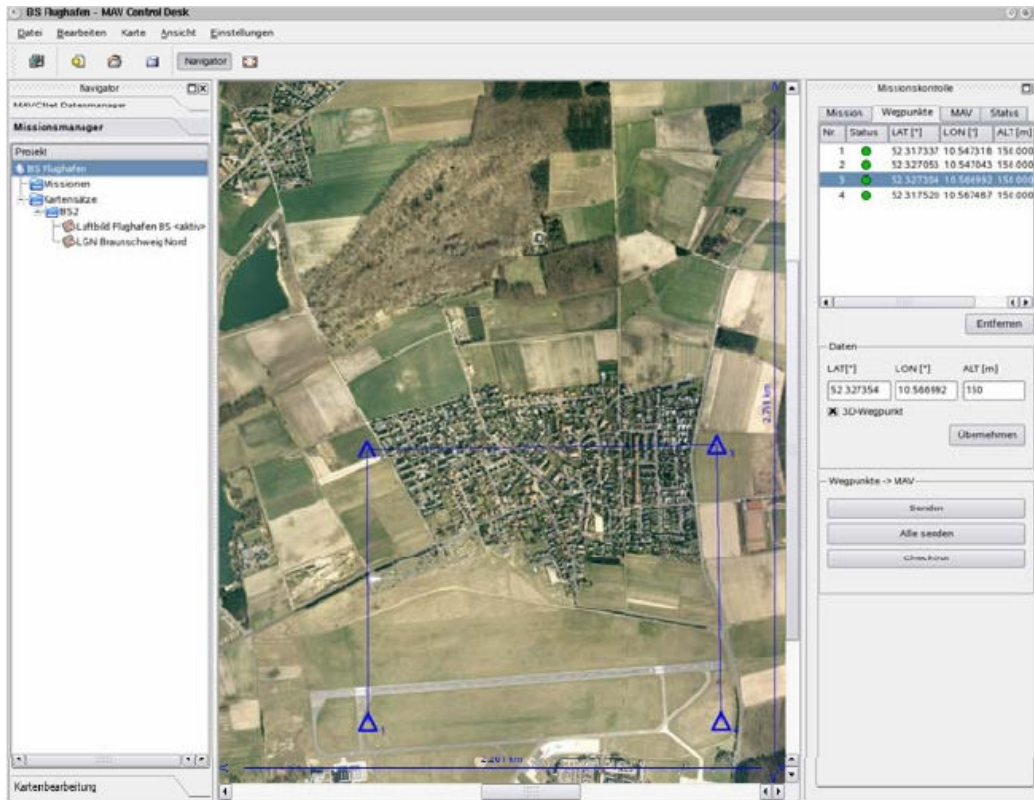
A data link is essential so that a planned mission can easily be transmitted to the UAV. Furthermore, the data link is essential in the relay of information back to the ground station. This information may be related to the UAV status or it might be a video feed back for reconnaissance purposes.

The implementation of a data link is generally in the form of a wireless serial or Ethernet radio modem. For more information please refer to Fernando (2006).

### ***2.4 Mavionics***

Mavionics is an example of a baseline Mission Planner. The software package allows 3D flight path way-point definition; with way-point actions as an option. Furthermore, the software package incorporates UAV monitoring. This is achieved via a position display on a user-defined map and system state information such as attitude, course and height. A screenshot of the Mavionics software package is shown in Figure 1.

However, this application does not assist the operator in planning missions. That is, it does not take into consideration some of the possible limitations that were listed in section 2.1. However, mission planning systems that perform such high level planning are military specific programs, which besides their astronomical costs also require fully equipped ground stations.



**Figure 1: Mavionics screenshot**

Therefore, it is proposed that a MPS be specifically built for the Monash UAV. Such an application should provide most of the functionality exhibited by the Mavionics package. Furthermore, the package needs to provide some of the desired features as presented by DSTO (1995); for instance, path planning for obstacle avoidance. This is discussed in the following Chapter.

### ***3.0 Path Planning***

UAVs are required to operate in environments with numerous threats present. This is especially true for military UAVs. Therefore, path planning is essential to a UAV system. This chapter presents several different approaches to path planning. The presented methods are evaluated for strengths and weaknesses, and an algorithm is selected for implementation of this component.

#### ***3.1 Path Planning Approaches***

The primary function of any path planning algorithm is to compute a trajectory from the UAVs present location to the target destination, whilst avoiding all obstacles. Path planning for a UAV can be categorised as static or dynamic. Static path planning requires an environment consisting of non-moving objects. In contrast, a dynamic path planning algorithm is suitable for situations when partial obstacle information is available and the environment is unpredictable and varies with time (Zelek, 1995). This is referred to as a dynamic environment. Since UAVs operate in such an environment, only dynamic path planning will be analysed in this thesis.

A dynamic path planning algorithm will require the path to be continually recomputed as additional obstacle information becomes available (Zelek, 1995). Therefore, the path planning algorithm must be computationally efficient. Furthermore, the path planning algorithm must ensure that the generated paths are of minimal length and satisfy any dynamic constraints of the UAV (Bortoff, 2000).

Conventional approaches treat UAV path planning as an optimisation problem. Such approaches usually employ graph theory in a two step process. Initially, a *polygonal path* is generated from a *Voronoi graph* by applying Dijkstra's algorithm (Jun & D'Andrea, 2000). This step is similar to a roadmap approach in robot path planning. The second step involves path refinement to

satisfy any set constraints. Generally, the path refinement process will be performed to meet a specific target application (Lee et al., 2003).

Bortoff (2000) aimed to create a stealthy path through a set of pre-defined static radars. He employed the two step approach as mentioned above. The graph solution (from step one) was used as the initial condition for the simulation of ordinary differential equations, which represented virtual masses in a virtual force. The simulation was performed to find a locally stable equilibrium solution, with the solution treated as the optimal path. However, this approach is only targeted for stealthy path planning, and is unsuitable for the purposes of this research. Furthermore, the path planning algorithm assumes a constant altitude for the UAV; thus, ignoring the advantages of a rotary UAVs unique flight capabilities.

### ***3.2 Extended Local Optimization Strategy (ELOS)***

The path planning methods discussed in the previous section are computationally intensive, and thus not suitable for a real-time application. Nevertheless, some path planning methods do achieve computational efficiency; however, this is at the cost of optimality in terms of path length (Akram et al., 2004). Furthermore, the traversal of the Voronoi graph can yield suboptimal trajectories (Jun & D'Andrea, 2000). A further drawback of a graph approach is its lack of dimensionality. To overcome this problem, Bortoff (2000) suggests a sequence of graphs be used. However, this approach remains computationally intensive, and the computational load grows quickly with an increase in the number of radar sites (Jun & D'Andrea, 2000).

The ELOS strategy attempts to overcome these insufficiencies by generating an optimal path based on the sensory data and the shortest distance to the target. Therefore, the optimisation objectives of the ELOS strategy are the distance travelled and collision avoidance. The generated path is locally

optimal with respect to the obstacles detected, and globally optimal with respect to the shortest distance calculated.

The algorithm consists of the following 7 steps:

1. Receive obstacle location and current position
2. Extract information on immediate neighbouring locations
3. Check for obstacles at each location
4. Calculate distance to target through each obstacle free location using the equilibrium formula
5. Determine shortest distance point
6. Update current position with new position
7. Check if new position equals target location; if not equal repeat above steps, otherwise stop

The ELOS strategy utilises grids instead of graphs for the representation of its environment. This approach allows the ELOS strategy to avoid the computationally intensive algorithms associated with graphs. Furthermore, this simplified representation enables efficient mapping of new objects and quicker regeneration of an optimal path.

Existing path planning research utilising ELOS has only considered a 2-dimensional representation of the environment (Davari et al., 2005). Therefore, a 3-dimensional path planning algorithm must also be implemented to take advantage of the helicopters' omni-directional capabilities.

## ***4.0 Navigation***

Navigation is a critical component of a UAV system. If lacking knowledge of the current position, the UAV can not determine how to reach the desired location. Furthermore, without reliable navigation, the UAV can not ascertain if it has arrived at the target location. The navigation sub-component is called by the mission controller for the purposes of path planning and tracking. Navigation in turn provides the mission controller with accurate position, velocity and attitude information.

This chapter introduces the concepts and equations behind an Inertial Navigation System (INS). Also, it explains GPS and presents possible methods of INS and GPS integration as part of the overall navigation component.

### ***4.1 Background***

Previous research has taken various approaches to navigation. Dixon and Henlich (1997) suggest dead reckoning, local or global beacons, landmark based navigation and map based navigation as viable options for a navigation system. Dollery (2001), as part of the initial research at Monash University, investigated dead reckoning methods and beacon implementations. Landmark and map based navigation systems were not considered, as they were deemed unsuitable for a low cost UAV platform. The research concluded that a local beacon system, which can upgrade to Global Positioning System (GPS), should be implemented to aid a dead reckoning navigation system. Since Dollery's recommendations, GPS hardware has been acquired and will be utilised as a reference tool to aid the dead reckoning system; thus, avoiding the need for a local beacon system.

The conventional approach to dead reckoning navigation systems for UAVs is via the use of sensor data from an Inertial Measurement Unit (IMU), as proposed by Dollery (2001). The IMU provides acceleration measurements

and angular rates relative to the three body axes using accelerometers and gyroscopes respectively. The helicopter position, velocity and attitude can be estimated using this data. A major drawback of an IMU implementation is that errors related to gyro drift can grow without bound; these errors are discussed in section 4.4. Hence, the GPS is utilised as a reference system, which minimizes dead reckoning errors via periodical position updates from the GPS.

The final element of the navigation sub-component, as proposed by Dollery (2001), is the heading compass. The heading compass is used to assist the yaw gyro similar to the way in which GPS assists position navigation. The yaw gyro, which helps stabilize yaw movement of the helicopter, has errors due to gyro drift. Thus, a heading compass is utilised to minimise this error. This mixed approach of dead reckoning and referencing strategies improves sensor redundancy, with one group of sensors compensating for the weaknesses of the other group of sensors (Dollery, 2001).

The use of inertial data from an IMU as a means of navigation is known as Inertial Navigation and is discussed in the next section.

## ***4.2 Inertial Navigation***

An Inertial Navigation System (INS) is a form of dead reckoning. It uses the accelerations and angular rates relative to the X, Y and Z body axes to determine the current position, velocity and attitude. The acceleration and angular rates are provided to the INS by the Inertial Measurement Unit (IMU).

The calculation of position, velocity and attitude is achieved through well known mathematical concepts. Acceleration is the derivative of velocity; therefore, by integrating the IMU reading for acceleration along an axis the velocity along that axis is derived (assuming initial velocity is known).

Similarly, integrating the velocity will give the change of position along that axis. Finally, it is possible to determine the current position if the direction of travel is known. This, of course, is a simple example of a 1 degree of freedom INS. In reality an INS will track 6 degrees of freedom, and thus is a far more complex problem. In addition to the increased degrees of freedom, noise and unbounded error problems need to be addressed for the INS to produce any meaningful results (Walchko & Mason, 2002).

There exist two types of Inertial Navigation Systems: Gimballed INS and Strap-down INS. In the last 20 years Strap-down systems have been prevalent as they reduce size, cost, power consumption and complexity of the system (Walchko & Mason 2002).

### *Gimballed Systems*

These were the first type of Inertial Navigation Systems. The accelerometers were mounted on a motorized gimballed platform. Thus, the accelerometers were isolated from the body rotations. A set of gyroscopes were employed to monitor the vehicles attitude. The motorized gimbals keep the accelerometers aligned to the navigation frame based on the attitude of the vehicle.

### *Strap-down Systems*

Strap-down systems are fixed (strapped) to the body of the vehicle. Thus, accelerations measured in the X, Y and Z axes are relative to the body frame. Therefore, a software solution is needed to keep track of the orientation of the IMU and vehicle. The IMU measurements are then rotated from the body frame to the navigation frame.

The reference frames required for a strap-down INS, as well as the transformation methods between the frames are described in the subsequent section.

## 4.2 Reference Frames and Transformations

### *Inertial frame (i frame):*

The inertial frame has an origin at the centre of Earth gravity with non-rotating axes. The X-axis points in the direction of the Vernal Equinox and the Z-axis is parallel to the north polar axis of the earth. An additional axis, the Y-axis, is added to make this a right-handed orthogonal co-ordinate system.

### *Earth frame (e frame):*

The earth frame also has an origin at the centre of mass of Earth and the Z-axis parallel to the rotation axis of the earth. The X-axis points towards the *Greenwich prime meridian* and the Y-axis completes the right handed orthogonal frame. Unlike the inertial frame, the earth frame rotates with Earth; thus, the inertial and earth frame longitudes differ as a linear function of time. This is depicted in Figure 2.

Earth frame longitude is measured east (+) and west (-) of the Greenwich meridian with latitude being measured north (+) and south (-) with respect to the equator.

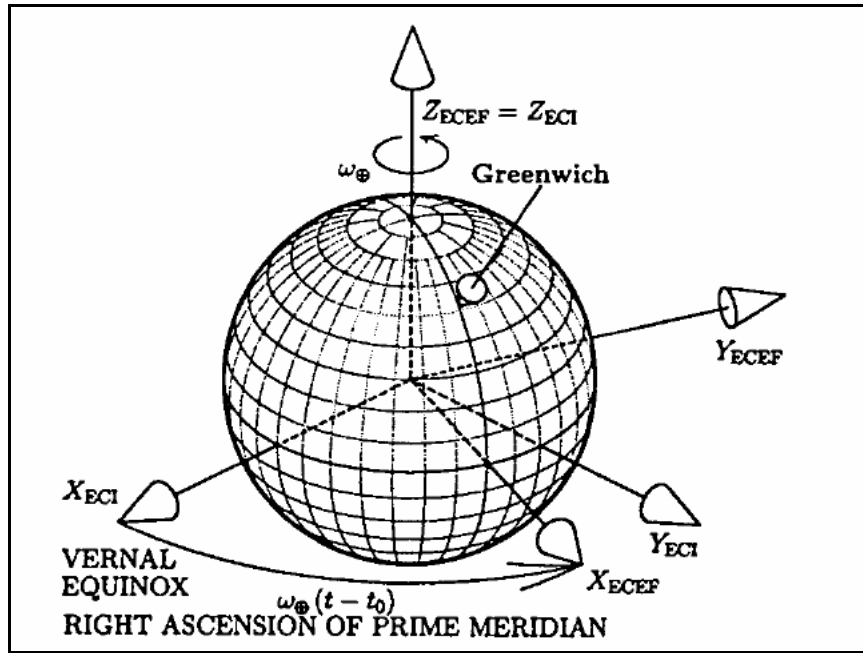


Figure 2: Inertial frame and earth frame coordinates comparison, from Grewal (2001)

*Navigation frame (n frame):*

The navigation frame is a locally level co-ordinate frame, which treats the earth as a flat surface. Its origin coincides with that of the sensor frame. There are two locally level co-ordinate systems employed; East-North-Up (ENU) and North-East-Down (NED). The ENU may be preferred because altitude increases in the upward direction. However, NED is more prevalent as its axes coincide with the vehicle fixed roll, pitch and yaw axes, when the vehicle is level and headed north (Grewal et al., 2001).

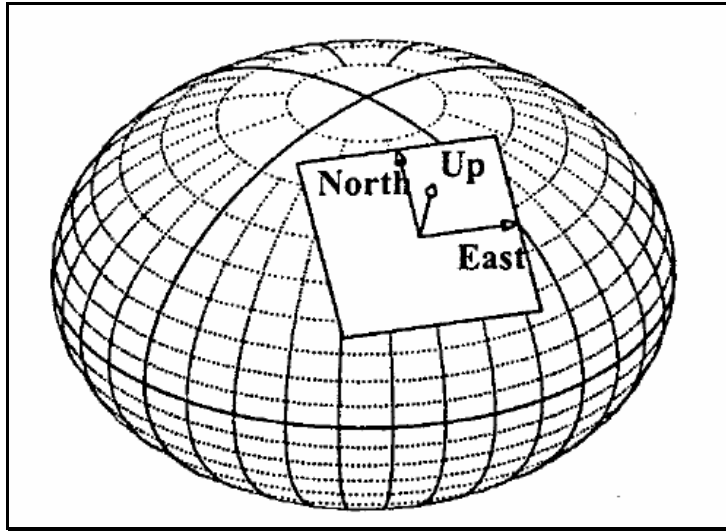


Figure 3: Navigation frame coordinates from Grewal et al. (2001)

*Body frame (b frame):*

The body frame is vehicle fixed. The roll, pitch and yaw axes are aligned with the roll, pitch and yaw of the vehicle as shown in Figure 4.

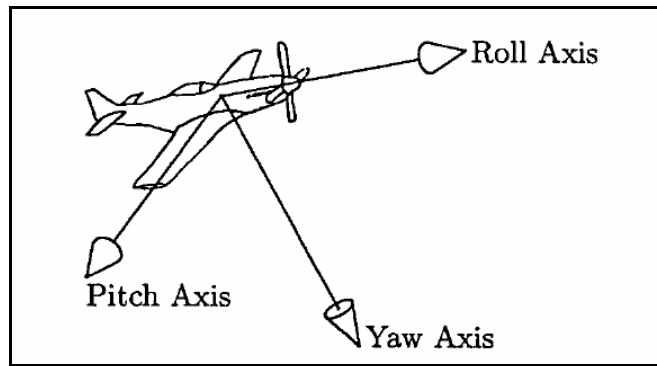


Figure 4: Body frame coordinates, from Grewal et al. (2001)

The transition between these frames is performed through the use of *Direction Cosine Matrices (DCM)*. The DCM from the e-frame to the n-frame is defined as:

$$C_e^n = \begin{bmatrix} -\sin \lambda \cos \mu & -\sin \lambda \sin \mu & \cos \lambda \\ -\sin \mu & \cos \mu & 0 \\ -\cos \lambda \cos \mu & -\cos \lambda \sin \mu & -\sin \lambda \end{bmatrix} \quad (4.1)$$

The DCM from the n-frame to the e-frame can then be obtained by using the orthogonality and is defined as:

$$C_n^e = (C_e^n)^T = \begin{bmatrix} -\sin \lambda \cos \mu & -\sin \mu & -\cos \lambda \cos \mu \\ -\sin \lambda \sin \mu & \cos \mu & -\cos \lambda \sin \mu \\ \cos \lambda & 0 & -\sin \lambda \end{bmatrix} \quad (4.2)$$

Titterton and Weston (1997) define the DCM from the n-frame to the b-frame as:

$$C_n^b = R_x(\phi)R_y(\theta)R_z(\psi) \quad (4.3)$$

where,  $\phi, \theta, \text{ and } \psi$  are the Euler angles of roll, pitch and yaw, respectively.

Thus, the DCM from the b-frame to the n-frame can be obtained by applying orthogonality:

$$\begin{aligned} C_b^n &= (C_n^b)^T \\ &= \begin{bmatrix} \cos \theta \cos \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \end{aligned} \quad (4.4)$$

### 4.3 Inertial Navigation Equations

The INS equations used to calculate position, velocity and attitude are relative to the navigation frame, which is oriented in the same pointing directions with respect to the local ground plane and the local gravity direction. This method, known as a navigation frame mechanisation, is appropriate for most on-board guidance systems as the flight control is usually carried out with respect to the local ground plane.

#### 4.3.1 Position and Velocity Equations

The accelerometer readings from the IMU,  $f^b$ , are measured in the body frame, and thus need to be converted to the navigation frame. This is achieved by using the  $b$  to  $n$ -frame DCM,  $C_b^n$ . Therefore, the accelerations in the navigation frame are defined as:

$$f^n = C_b^n f^b \quad (4.5)$$

Furthermore, the INS must account for external accelerations which are not measured by the IMU. These include *Coriolis acceleration*, *centripetal acceleration* and *gravitational force*. Therefore, the equation for rate of change of velocity in the navigation frame is defined as:

$$\dot{V}^n = f^n - (\omega_{in}^n + \omega_{ie}^n) \times V_e^n + g_l^n \quad (4.6)$$

Where  $\omega_{in}^n$ , is the sum of the Earth rotation rate and the rotation rate of the navigation frame with respect to the fixed earth frame. Therefore,  $\omega_{in}^n$  can be defined as:

$$\omega_{in}^n = \omega_{ie}^n + \omega_{en}^n \quad (4.7)$$

The term  $\omega_{ie}^n$  is the earth rotation rate,  $\omega_e$ , projected on to the navigation frame. It can be defined as:

$$\omega_{ie}^n = (\omega_e \cos \lambda, 0, -\omega_e \sin \lambda)^T \quad (4.8)$$

The rotation rate of the navigation frame with respect to the fixed earth frame,  $\omega_{en}^n$  (from equation 4.7), is also known as the transport rate. It is defined as:

$$\omega_{en}^n = ( v_E / (N + h), -v_N / (M + h), -v_E \tan \lambda / (N + h) )^T \quad (4.9)$$

where  $v_E$  and  $v_N$  are the velocities in the east and north direction, respectively.  $h$  is defined as the altitude of the vehicle and  $M$ ,  $N$  are the meridional radius of curvature and transverse radius of curvature. The formula for calculating  $M$  and  $N$  is presented in Appendix A.

By substituting equation 4.7 into 4.6, a final form of the equation for the rate of change of velocity is determined:

$$\dot{\mathcal{V}} = f^n - (2\omega_{ie}^n + \omega_{en}^n) \times V_e^n + g_l^n \quad (4.10)$$

Where  $V_e^n$  is the velocity vector and  $g_l^n$  is the term combining acceleration due to gravity and the centripetal acceleration. The equation for  $g_l^n$  is defined in Appendix A.

Equation 4.10 is integrated to determine the aircrafts velocity, and the velocity is integrated to obtain the current position of the aircraft.

### 4.3.2 Attitude Equations

The attitude of the aircraft is represented by the roll, pitch and yaw angles. The angles represent the orientation of the aircraft relative to the navigation frame, and are used to determine the values of the DCM for b to n-frame conversion,  $C_b^n$ . The propagation of  $C_b^n$  can be defined as:

$$\dot{C}_b^n = C_b^n (\omega_{nb}^b \times) = C_b^n ((\omega_{ib}^b - \omega_{in}^b) \times) \quad (4.11)$$

where  $\omega_{ib}^b$  is the vector of gyroscope readings from the IMU and  $\omega_{in}^b$  is the sum of the earth rotation rate and the transport rate,  $\omega_{in}^n$ , projected onto the body frame. Therefore,  $\omega_{in}^b$  can be defined as:

$$\omega_{in}^b = C_n^b \omega_{in}^n \quad (4.12)$$

A block diagram, demonstrating the inter-relations of the equations of the navigation frame mechanisation, is shown in Figure 5 below.

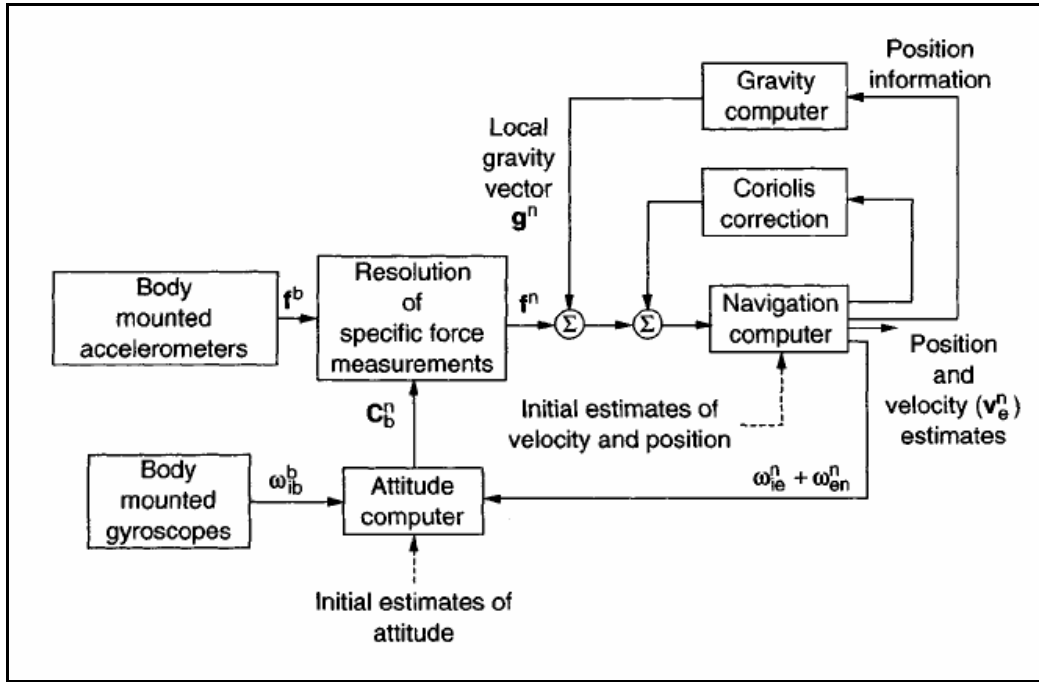


Figure 5: INS, navigation mechanisation block diagram, from Titterton and Weston (1997)

#### 4.4 Inertial Navigation Error Sources

Like any dead reckoning navigation method, the INS is prone to error. The majority of INS errors can be contributed to the measurements provided by the IMU. However, other factors such as initialisation and gravity calculation can also have dramatic effects on the accuracy of the INS. Table 1 presents the major causes of INS inaccuracies.

Error Source	Description
Accelerometer bias	A constant offset in the accelerometer output
Accelerometer scale factor	Acceleration error proportional to sensed acceleration
Gyroscope drift	A constant gyroscope output even when no angular rate present
Gyroscope scale factor	Angular rate error proportional to sensed angular rate
Nonorthogonality of axes	Misalignment of the accelerometer and gyroscope axes
Temperature	Gyroscope drift due to temperature change
Initial misalignment	Inaccurate initialisation of roll, pitch and yaw
Gravity anomalies	Changes in gravity which are not modelled in the gravity calculation

Table 1: INS errors, adapted from Kumar (2004)

#### 4.5 GPS

The Global Positioning System is satellite based navigation which was developed by the U.S. Department of Defence. GPS was declared operational in 1994 and currently consists of 24 satellites.

The system provides data of high absolute and consistent accuracy (Cramer, 1997). It uses the time difference between transmission time of a signal from a satellite and the receiving time of the GPS receiver to obtain the range between receiver and satellite. This time difference is known as the pseudorange. To obtain accurate position information, the receiver uses at least four different pseudoranges.

However, GPS has errors relating to clock offsets between the satellite and receiver which affect GPS accuracy. Furthermore, GPS can have outages; this occurs when the GPS receiver does not have access to four satellites to determine the pseudoranges. Therefore, GPS navigation requires the aid of inertial navigation; just as much as inertial navigation needs the aid of GPS.

#### **4.5.1 INS/GPS Integration**

The integration of INS and GPS is conventionally performed through the use of a Kalman filter. A Kalman filter is a stochastic estimator which is used to predict the INS errors. A detailed discussion on the Kalman filter is performed in the next section.

The INS position, velocity and attitude information is corrected using the Kalman filter error estimates. There exist two methods for error correction: feed forward and feed back. In a feed-forward error correction setup, the INS assumes that there is no correction, and thus the errors of the INS continue to grow without bound. In contrast, the feed-back scheme continually corrects the INS thus keeping INS errors relatively small. However, both methods eventually result in the same output; therefore, this is purely a design issue for GPS/INS integration.

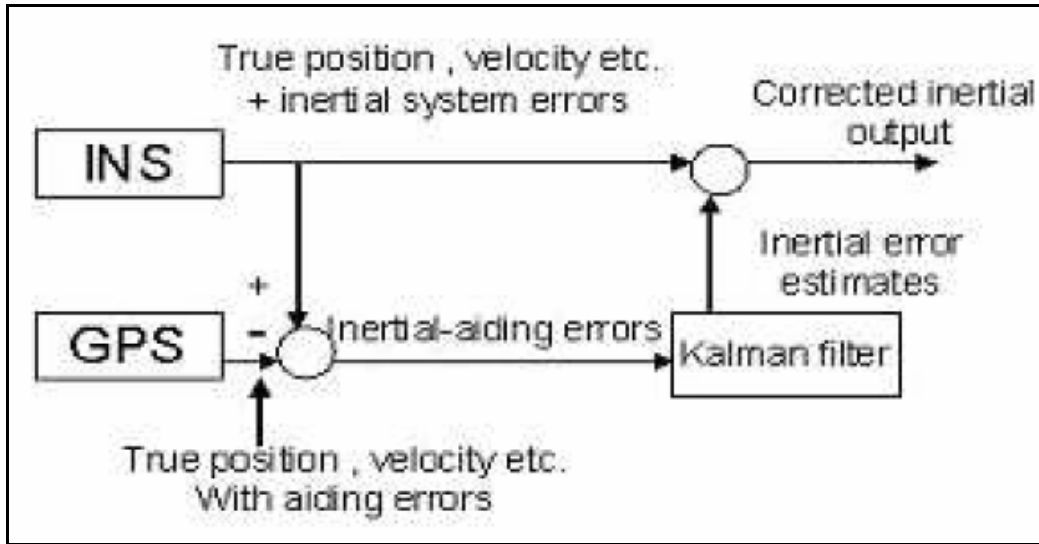


Figure 6: Feed-forward aided INS, from Kumar (2004)

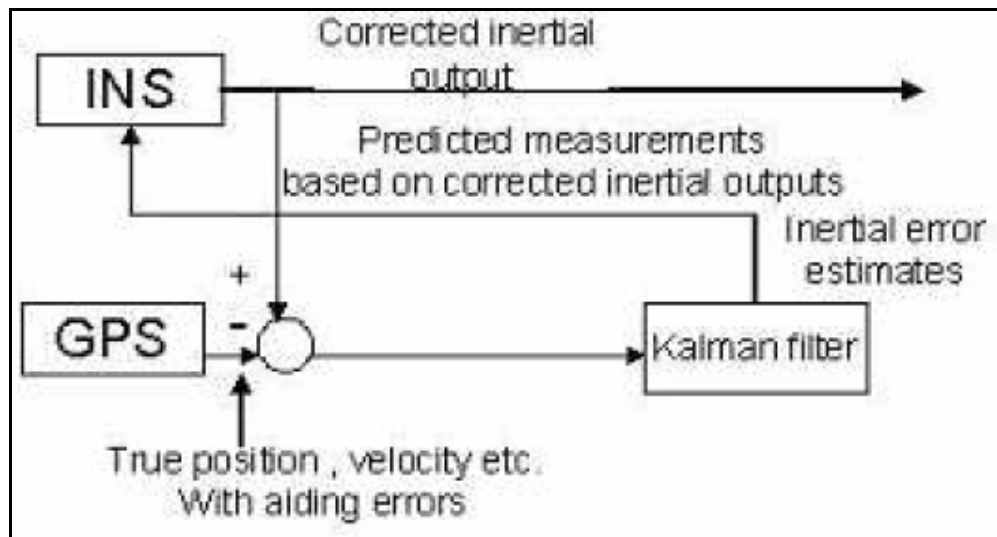


Figure 7: Feed-back aided INS, from Kumar (2004)

#### 4.6 Kalman Filter

The Kalman filter is an integral component of a navigation system. It is utilised for convenient integration of navigation sensors in order to achieve optimal system performance. The filter utilises statistical models to weight new measurements relative to previous information, and thus produce current estimates of system variables. The use of Kalman filters is extremely

popular in the integration of INS and GPS, largely due to its optimum performance.

Previous work on INS and GPS integration has produced varying results; Hide and Moore (2005) implemented a 15 state filter, while Wolf et al. (1997) designed a 27 state Kalman filter. Conventionally the Kalman filter states are the error estimates for position, velocity, attitude and possibly accelerometer and gyroscope biases. However, Li et al. (2005) concluded that a 15 state Kalman filter, estimating the sensor biases, produced marginally better results when compared to a 9 state Kalman filter with constant sensor bias estimates. Furthermore, Minkler & Minkler (1993) suggest that implementation of a 9 state filter should be performed before any attempts at larger filters are made.

#### 4.6.1 Kalman Filter Equations

The Kalman filter optimally estimates the states of a system, based on noisy measurements received from the system. In addition, the estimation can be performed in real time, as the filter is in the form of a computationally efficient, recursive algorithm (Minkler & Minkler, 1993). The states of a system describe its behaviour as a function of time. For example, position, velocity and attitude can be states of a system. The noisy state measurements can be thought of as the desired signal plus noise (Levy, 1997). Therefore, the Kalman filter is utilised to estimate a statistically optimal value of the desired signal. It is optimal in that the mean-square estimation error is minimised.

For systems described by a linear state space model:

$$x_{k+1} = \Phi_k x_k + G_k w_k \quad (4.13)$$

the Kalman filter provides an optimal mean-square error system state,  $x_k$ , estimate based on measurements  $z_k$ ; defined as:

$$z_k = H_k x_k + v_k \quad (4.14)$$

where,

-  $x_k$  is an  $n$ -dimensional system state vector at the  $k$ -th time point

- $z_k$  is an  $m$ -dimensional vector representing the measurement or output of the system
- $w_k$  is a  $p$ -dimensional vector representing random system process noise
- $v_k$  is a  $m$ -dimensional vector representing random measurement noise
- $\Phi_k$  is the  $n$  by  $n$  non-singular state transition matrix at the  $k$ -th time point
- $G_k$  is an  $n$  by  $p$  system associated matrix at time point  $k$
- $H_k$  is the  $m$  by  $n$  observation matrix at time point  $k$

It is assumed that the system process and measurement noise vectors,  $w_k$  and  $v_k$ , are independent of each other, white, and have normal probability distributions. If these assumptions hold we can define the covariance functions of  $w_k$  and  $v_k$  as:

$$\begin{aligned} E(w_k w_k^T) &= Q_k \\ E(v_k v_k^T) &= R_k \end{aligned}$$

where,  $Q_k$  and  $R_k$  are known  $p$  by  $p$  and  $m$  by  $m$  symmetric positive semi-definite matrices, respectively, and  $R_k$  is non-singular.

Based on the knowledge of the process, an initial estimate of the process state, denoted as  $\hat{x}_k^-$ , and the associated covariance matrix,  $P_k^-$ , is provided. The Kalman filter attempts to improve the estimate using a measurement  $z_k$ . To achieve this, the Kalman filter first calculates the Kalman gain. The Kalman gain is the blending factor which reduces the mean-square estimation error. It is defined as:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad (4.15)$$

The system estimate and the associated error covariance matrix are then updated using the following equations.

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H_k \hat{x}_k^-) \quad (4.16)$$

$$P_k = (I - K_k H_k) P_k^- \quad (4.17)$$

Finally, the next step measurement  $\hat{x}_{k+1}^-$ , and its error covariance  $P_{k+1}^-$  are estimated, using the following equations, before the process is repeated.

$$\hat{x}_{k+1}^- = \Phi_k \hat{x}_k \quad (4.18)$$

$$P_{k+1}^- = \Phi_k P_k \Phi_k^T + Q_k \quad (4.19)$$

The algorithm is presented in its recursive form in the figure below.

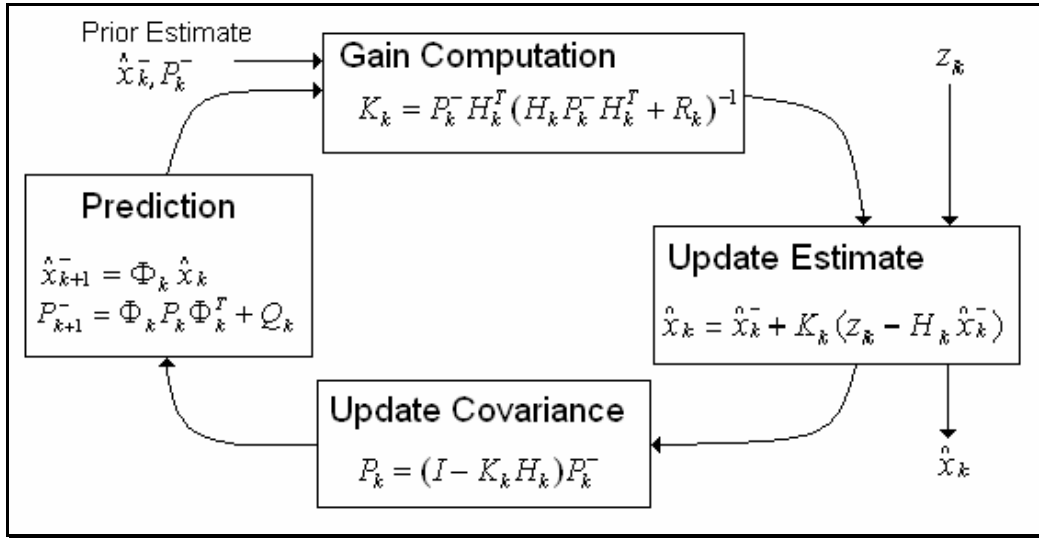


Figure 8: The Kalman Filter loop

#### 4.7 Summary

This Chapter has presented an outline of the equations required to implement a Strap-down Inertial Navigation System. The process requires the tracking of aircraft orientation with respect to the navigation frame; this is achieved via gyroscope readings from the IMU (after correction for Earth rotation rate and the transport rate). Using the orientation information, the accelerometer readings from the IMU are rotated to the navigation frame. After correcting the accelerometer readings for external accelerations, such as Coriolis and gravity, an integration of the acceleration values produces the aircraft

velocities. A further integration of the velocities produces the aircrafts position. However, a referencing aid such as GPS is required to correct for errors in position, velocity and attitude calculations. The integration of the two entities is performed via a Kalman filter. The following Chapter discusses the implementation of the above procedure.

## ***5.0 Implementation***

This Chapter discusses the implementation of the required sub-components for the UAV. The Mission Planning System is discussed followed by the path planning algorithm and its extensions. Finally, the implementation of the Inertial Navigation and its integration with GPS is presented.

### ***5.1 OHBY Mission Planning System***

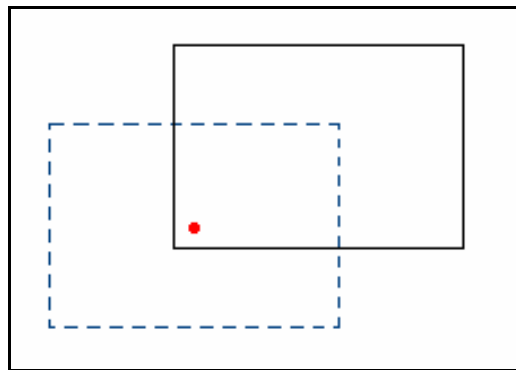
Many of the required or desired features of a Mission Planning System, as presented in Chapter 2, are out of the scope for this research. As with the development of any large scale program, a base application which meets some critical requirements has been initially developed. To accomplish this, a list of the critical requirements of a MPS was created. This list includes the following:

1. The ability to add, remove and modify way-points in the mission list
2. The ability to insert any known static objects
3. These objects must also be removable
4. The user should be able to plot the expected flight path
5. The user should be able to run the path planner to solve a flight path
6. The user must be able to modify the solved path

These elements were developed with the intention to provide a basic mission planner for future work. The basic features, listed above, should suffice for

the purposes of entering a UAV competition such as the International Aerial Robotics Competition (IARC).

The developed system, OHBY-MPS, has sufficient functionality to allow the Monash UAV to be entered into competitions such as the IARC. The system clearly defines the mission boundaries, restricting the UAV mission area to a 2 km by 2 km flight envelope. The initial UAV position is centred inside this flight envelope. However, the user is able to translate these boundaries so that the initial UAV position is located anywhere inside the flight envelope. This is demonstrated in Figure 9, where the dashed square represents the initial flight envelope and the solid square is the new flight envelope.



**Figure 9: Flight envelope translation**

The mission plan is stored as a linked list, with each node inside the list representing a way-point that must be visited by the UAV. The structure of the way-point node is shown in Table 2.

Data Type	Name	Purpose
int	id	Stores the way-point ID
float	lat	Latitude co-ordinates of way-point
float	lon	Longitude co-ordinates of way-point
float	alt	Altitude of way-point
Task_type	task	Task to be performed at way-point

**Table 2: Way-point structure**

The task type is employed to allocate a task to be performed by the UAV at a given way-point. At present, the possible choices include none, hover, search for building, locate window or loiter. This is just a preliminary list of tasks which would be required to enter the IARC competition. However, this list is greatly dependant on the UAV capabilities. A highly advanced UAV will be able to perform tasks far more complex then those listed, as well as a greater variety of tasks.

The mission file which is sent to the UAV has the following form:

- Line 1: Mission Boundaries – Upper and Lower latitude/longitude boundaries
- Line 2: Way-Point 0 – initial location
- Line 3: Way-Point 1
- Line 4: Way-Point 2

### ***Mission Planning***

OHBY enables basic mission editing. This includes the addition, insertion, removal or modification of way-points which form the mission list. Known static objects which are inside the flight envelope can also be entered by the operator. To envisage a representation of the expected flight the user can “*plot path*”. This option is similar to connect the dots. Alternatively, the user may wish to solve the path; that is, OHBY will run the implemented path planner to generate an optimal collision-free path. The solved path is displayed and further modification, if required, is possible. Currently, the path planner implemented is the ELOS 2D algorithm. The commands *Start mission*, *Stop*

*Mission, Continue Mission and Cancel Mission* are used to execute the planned mission. Their operations are explained in Table 3.

<b>Command</b>	<b>Action</b>
Start mission	<ul style="list-style-type: none"> <li>- Saves the mission list as a file</li> <li>- Sends the created file to UAV</li> </ul>
Stop mission	<ul style="list-style-type: none"> <li>- Sends a stop signal to UAV</li> <li>- Receives from the UAV a list of way-points not yet visited</li> </ul>
Continue mission	<ul style="list-style-type: none"> <li>- Saves the edited mission as a file</li> <li>- Sends the file to the UAV</li> <li>- If mission is unedited it sends a continue signal instead</li> </ul>
Cancel mission	<ul style="list-style-type: none"> <li>- Sends a cancel mission signal to UAV</li> <li>- Optional return to base signal is available</li> </ul>

**Table 3: Mission commands**

Furthermore, a structure for manual control of the UAV has been implemented. However, due to the incompleteness of the flight control algorithm by the Engineering department, it is unfeasible to completely implement such a feature. The manual controls feature of OHBY will allow untrained operators to control the UAV. Currently, only a trained pilot can fly the helicopter/UAV. The operations that are implemented include fly forward, hover and turn left/right. These operations will prove to be adequate for data collection or testing of new sensors, without the need for a trained pilot.

### ***Mission Controller***

Similar to the OHBY user interface the mission controller is still in a primitive stage. The mission controller, which will be deployed on-board the UAV, receives sensor information on its immediate environment through a simulated on-board camera. When an object is detected, the mission controller

sends the current location (from simulated navigation) and the object locations (from simulated on-board vision system) to the path planner. The path planner then determines an optimal path around the obstacle and returns control back to the mission controller. The mission controller subsequently resumes the mission. Figure 10 illustrates the communication channels of the mission controller.

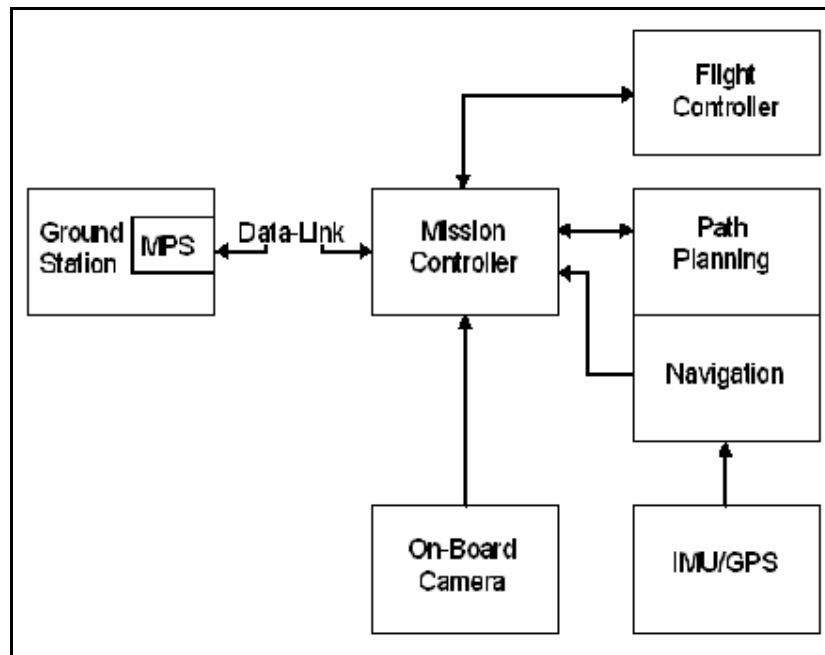


Figure 10: Mission controller interaction diagram

### 5.1.1 Graphical User Interface

The graphical user interface was developed using Glade-2. Glade-2 is practical for building user interfaces for GTK+ and Gnome applications. It can be used with any desktop environment in Linux, which has the GTK and Gnome libraries installed. The user interface was implemented using Glade as it is assumed the mission planning will be accomplished using the Linux operating system, which is stable and similar to QNX; the real-time operating system onboard the UAV.

Figure 11, shown below, presents the main screen of the mission planning system. OHBY has a simplistic design aimed to reduce the complexity of mission planning. The way-points in the main screen are displayed as red dots and objects as black dots. In addition, all the current options are presented in a side menu bar allowing easier access.

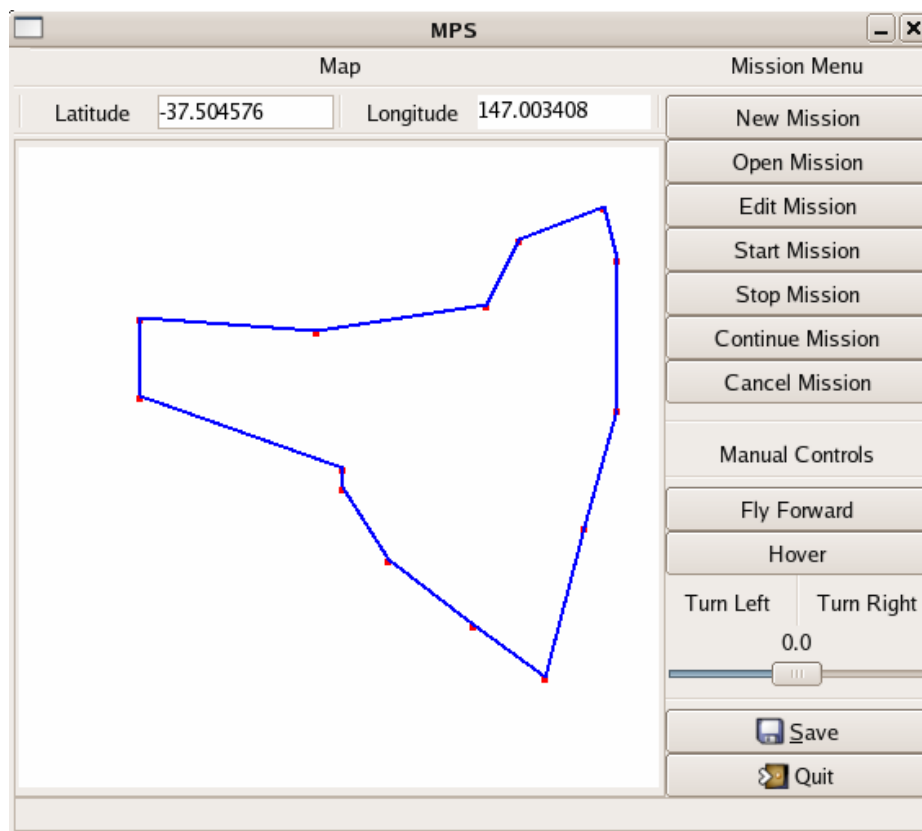


Figure 11: OHBY Main screen

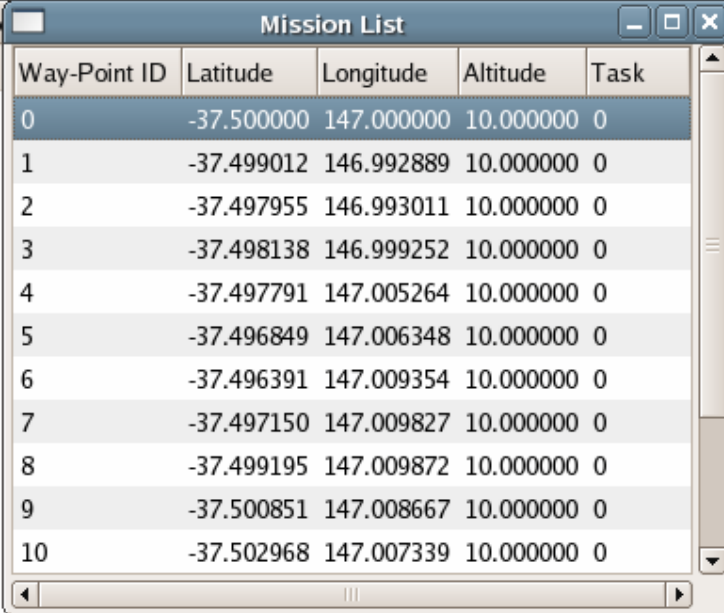
The map, which may seem simplistic, is used to display way-points, objects and the path between way-points. Furthermore, the GUI enables interactive input of object and way-point data via the map; the methods are explained in Table 4. Also, latitude and longitude coordinates of a pointer over the map are displayed to assist in this process.

<i>Task</i>	<i>Action</i>
Add Object	Click middle-button on mouse
Remove Object	Click right-button
Add Way-Point	Click left-button
Remove Way-Point	Double-click right button

**Table 4: Interactive object and way-point input**

In addition to the main screen, a second screen is displayed when a mission is being planned. This screen is known as the mission list. It displays the details of the way-points in the current mission list in its order of execution.

Therefore, the user is able to identify a way-point through the mission list and take a course of action such as removal, modification or insertion, without significant delay or hassle. Figure 12, shows the mission list window.



Way-Point ID	Latitude	Longitude	Altitude	Task
0	-37.500000	147.000000	10.000000	0
1	-37.499012	146.992889	10.000000	0
2	-37.497955	146.993011	10.000000	0
3	-37.498138	146.999252	10.000000	0
4	-37.497791	147.005264	10.000000	0
5	-37.496849	147.006348	10.000000	0
6	-37.496391	147.009354	10.000000	0
7	-37.497150	147.009827	10.000000	0
8	-37.499195	147.009872	10.000000	0
9	-37.500851	147.008667	10.000000	0
10	-37.502968	147.007339	10.000000	0

Figure 12: Mission List display

The edit mission menu, shown in Figure 13, is the central piece of the mission planning process. The menu provides options for editing the way-points comprising a mission. These tasks include addition, removal, modification, and insertion of way-points as well as the translation of flight boundaries. In addition, the user is able to edit object information or solve a path based on the object information. Thus, the user can customise the mission in any way they like, providing a valid path exists.



Figure 13: Edit Mission menu

## 5.2 ELOS Path Planning

The steps involved in the ELOS path planning strategy were presented in Chapter 3. This section discusses the implementation of the algorithm. To begin, a 2-dimensional path planner was implemented for this component. Once the algorithm was verified, the path planner was extended into a 3-dimensional optimal path planner. In addition, further work was carried out, to provide mission specific features; specifically, the implementation of no-fly zones.

### 5.2.1 ELOS 2D Path Planning

In the ELOS 2D algorithm a matrix is utilised to represent the area of navigation. For ease of interpretation, the generated path is stored in a path matrix while obstacles are mapped onto an obstacle matrix.

The path planner implemented as part of this research requires a start and target location to begin path generation. Upper and lower boundaries for the latitude and longitude are provided by the mission controller based on the mission plan. These boundaries represent the  $0, 0$  and  $n, m$  positions of the matrices, where  $n$  is the width and  $m$  is the height of the matrix. Using the provided latitude and longitude boundaries the algorithm can map the start location and target location onto the path matrix.

The path planner receives obstacle information from the mission controller, based on the sensor readings. Any obstacles present are mapped into the obstacle matrix. However, as no vision system was available this aspect of the path planning process was performed via a simulation program. The algorithm then proceeds to extract the obstacle information of the neighbouring cells. For each obstacle free cell a distance relative to the target is calculated using the equilibrium formula shown below:

$$dist = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} + \sqrt{(x_t - x_c)^2 + (y_t - y_c)^2} \quad (5.1)$$

where,  $x_s, y_s$  is the start location,  $x_t, y_t$  is the target location and  $x_c, y_c$  is the current location. As the algorithm checks all the obstacle free paths, it tracks the minimum distance and the direction of the new cell relative to the current cell. The minimum distance position is selected and set as the current location. If the current location equals the target location the path planner terminates, otherwise it repeats this process.

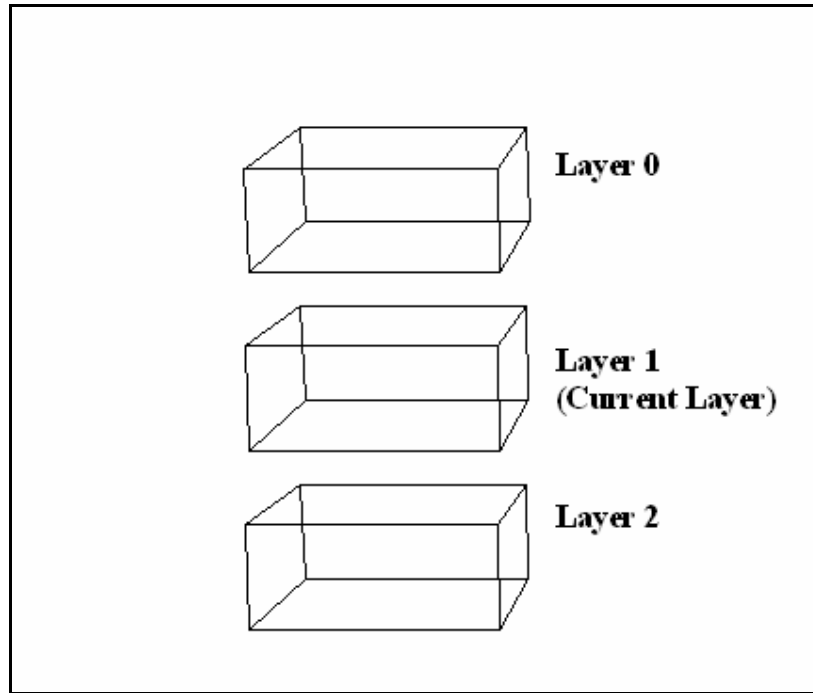
Furthermore, the algorithm is applied twice, to improve the optimality of the path generated. The first run applies the algorithm from start location to target location; the second run reverses this and determines a path from target location to start location. A comparison is then made on the number of steps of each path to determine the shortest path. If the paths are of equal length a *path time calculator* is utilised. The path time calculator traverses the path and increments an internal counter once for every step made and once more if a turn is made. The path times of each generated path are compared and the quickest path is selected. Currently, this process logically runs only when paths are of equal length; however, if desired, a path time calculation could be performed after every path generation.

### **5.2.2 ELOS 3D Path Planning**

To exploit the rotary UAVs unique flight capabilities, it was essential that the path planner be extended to perform 3-dimensional mapping. To achieve this, the following modifications had to be made to the 2D path planner:

- 3D representation of the navigation region
- Neighbouring cell extraction
- The distance formula needed to incorporate the new axis

Consequently, the representation of the navigation region was altered from a 2D matrix to a 3D cube. The extraction of neighbouring cells in a cube used the same approach as the 2D version, except this time it was necessary to extract neighbours from three layers; the current layer, the layer above and the layer below. This is demonstrated in Figure 14.



**Figure 14: 3D neighbourhood extraction**

The final modification required to implement the 3D path planner was the distance calculation. Calculation between two points in the 2D planner utilises the Pythagoras theorem, with the difference in x and y coordinates as input. In a 3D environment a simple extension is necessary. Initially, the Pythagoras theorem was applied as in the 2D system; the Pythagoras theorem is then applied once more. However, in this case, the inputs are the difference in z values and the result of the previous calculation. The end result is the distance between two 3D way-points. The formula is given in equation 5.2 below.

$$dist = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2 + (z_s - z_c)^2} + \sqrt{(x_t - x_c)^2 + (y_t - y_c)^2 + (z_t - z_c)^2} \quad (5.2)$$

This extension of the ELOS 2D algorithm has enabled the path planner to operate in a 3D environment. Thus, the UAVs flight abilities are taken full advantage of. However, this increased mobility also increases the requirements of the sensor system. The vision system, if implemented, must

detect any objects above and below the UAV to ensure a collision free path is indeed generated.

### 5.2.3 Threats and No Fly Zones

Besides objects, the UAV may have to contend with other flight restrictions. In military situations this may mean threat zones, where the UAV may come under attack. Therefore, this problem must be addressed by the path planner.

The extension of the path planning algorithm implemented includes *no-fly zones* for the UAV. The need for no-fly zones arose as a result of UAV competition requirements. Generally, a UAV competition will require that a UAV remain out of a specified no-fly zone. Furthermore, operators of the UAV may wish for the UAV to remain below a certain altitude. This can be achieved by specifying a no-fly zone at that altitude.

A possible implementation of this feature is to treat the off limit areas as objects. However, in some circumstances it may be essential for a UAV to enter an off limit area. For example, the UAV may be trapped and the only exit path is through a no-fly zone. Therefore, the implementation of this path planning extension is carried out via a third *cube* that is used to handle threats.

In order to enter a no-fly zone into the path planner, the operator is required to specify the latitude, longitude and altitude boundaries of the area to be delimited. In addition, it is required that a level of threat/restriction be specified. This feature is used to distinguish between strict no-fly zones and preferred flight paths.

The threat level is then assigned to the specified coordinates of the threat cube. The threat cube is utilised when calculating the shortest distance

through obstacle free cells. The threat level of a cell is added to the distance calculated. Thus, the total cost of a cell can be defined as:

$$CostTotal = dist + threat$$

### **5.3 INS module**

The INS module was implemented based on the equations presented in Chapter 4. This section presents the implementation of the aforementioned equations, with a detailed discussion on position, velocity and attitude integration and the equations used to perform these tasks.

#### **5.3.1 Inertial Measurement Unit**

The Inertial Measurement Unit currently on-board the UAV is the Phoenix O-Navi. It has a local coordinate set of East, North, Up and comes combined with a GPS receiver. The data from the IMU is streamed through the serial port. The data is available in two formats; ASCII and binary. The binary format allows a higher sampling rate and is, therefore, the preferred mode of operation.

However, the IMU did not come with software to allow data collection. The recommended approach for familiarisation was via the HyperTerminal in Windows. Therefore, to allow real-time data collection and analyses a program which runs under POSIX-standard operating systems was developed. Two separate programs, one for ASCII and the other for binary format were developed.

Significant hurdles were encountered during development due to inadequate and inaccurate specifications; this is further discussed in the Results and Discussion Chapter.

### 5.3.1 Attitude Integration

Attitude representation is conventionally performed via the use of quaternions. This is due to the efficient implementation of quaternions as they require no trigonometry functions (Farrell and Barth, 1998). Quaternions are an extension of complex numbers. They consist of four parameters, and are virtually a four element vector. The elements of a quaternion are  $q_1$ ,  $q_2$ ,  $q_3$  and  $q_4$ .

Before attitude integration is performed, it is essential that the gyro readings be adjusted to account for the earth rotation rate as defined in section 4.3.1. This angular rates adjustment is performed with the following equation:

$$\Delta\omega_{nb}^b = \Delta\omega_{ib}^b - C_n^b (\omega_{ie}^n + \omega_{en}^n) \Delta t \quad (5.3)$$

where,  $\Delta t$  is the time increment of the measurement and is equal to the inverse of the IMU sampling time. Using the magnitude of the angular increment,  $\Delta\omega$ , and the angular increments themselves, an update of the quaternions can be performed. This is achieved with the following equation:

$$q_{k+1} = q_k + 0.5 \begin{bmatrix} c & s\Delta\omega_x & -s\Delta\omega_y & s\Delta\omega_z \\ -s\Delta\omega_x & c & s\Delta\omega_x & s\Delta\omega_y \\ s\Delta\omega_y & -s\Delta\omega_x & c & s\Delta\omega_z \\ -s\Delta\omega_x & -s\Delta\omega_y & -s\Delta\omega_z & c \end{bmatrix} q_k \quad (5.4)$$

where  $s$  and  $c$  are defined as:

$$s = \frac{2}{\Delta\omega} \sin \frac{\Delta\omega}{2}$$

$$c = 2(\cos \frac{\Delta\omega}{2} - 1)$$

and the angular magnitude is defined as:

$$\Delta\omega = \sqrt{\Delta\omega_x^2 + \Delta\omega_y^2 + \Delta\omega_z^2}$$

Subsequently, the body to navigation DCM is updated using the following equation:

$$C_b^n = \begin{bmatrix} (q_1^2 - q_2^2 - q_3^2 + q_4^2) & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 + q_3q_4) & (q_2^2 - q_1^2 - q_3^2 + q_4^2) & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & (q_3^2 - q_1^2 - q_2^2 + q_4^2) \end{bmatrix} \quad (5.5)$$

If necessary, the Euler angles, which are the roll, pitch and yaw of the aircraft, can be extracted from  $C_b^n$  using the following equations:

$$\theta = -\tan^{-1}\left(\frac{c_{31}}{\sqrt{1-c_{31}^2}}\right) \quad (5.6)$$

$$\phi = a \tan 2(c_{32}, c_{33}) \quad (5.7)$$

$$\psi = a \tan 2(c_{21}, c_{11}) \quad (5.8)$$

where,  $c_{ij}$  is the (i,j-th) element of  $C_b^n$ . Alternatively, the system may only use quaternions for attitude representation. However, in the implementation of this INS, Euler angles and quaternions have been employed. The Euler angles

provide a visually superior representation of the attitude and are required for integration with the Kalman filter.

### 5.3.2 Velocity and Position Integration

The velocity and position integration begins with the translation of the accelerometer readings from body to navigation frame. Equation 5.9 performs this translation. Furthermore, it also applies first order sculling correction.

$$\Delta f^n = C_b^n \begin{bmatrix} 1 & 0.5\Delta\omega_x & -0.5\Delta\omega_y \\ -0.5\Delta\omega_x & 1 & 0.5\Delta\omega_x \\ 0.5\Delta\omega_y & -0.5\Delta\omega_x & 1 \end{bmatrix} \Delta f^b \quad (5.9)$$

The velocity increments are obtained after applying Coriolis and gravity correction:

$$\Delta v^n = \Delta f^n - (2\omega_{ie}^n + \omega_{en}^n) \times V_e^n \Delta t + g_l^n \Delta t \quad (5.10)$$

Velocity integration can be performed simply as:

$$v_{k+1}^n = v_k^n + \Delta v_{k+1}^n \quad (5.11)$$

Finally, the positions are integrated using the second order Runge-Kutta method:

$$r_{k+h}^n = r_k^n + 0.5 \begin{bmatrix} \frac{1}{M+h} & 0 & 0 \\ 0 & \frac{1}{(N+h)\cos\lambda} & 0 \\ 0 & 0 & -1 \end{bmatrix} (v_k^n + v_{k+h}^n) \Delta t \quad (5.12)$$

### 5.3.3 Summary

The INS module initially calculates the transport rate and earth to navigation frame DCM, based on current position. The earth rotation rate is then projected onto the navigation frame by  $C_e^n$ . Subsequently, the earth rotation rate and the transport rate are combined and projected onto the body frame. This value is then utilised to adjust the gyro readings from the IMU. Finally, the body to navigation frame DCM is updated, as explained in section 5.3.1.

The acceleration readings from the IMU are adjusted using the updated DCM and the gravity calculation. The adjusted readings are then integrated to obtain velocity, and velocity is integrated to obtain current position.

The figure below depicts this process in a block diagram.

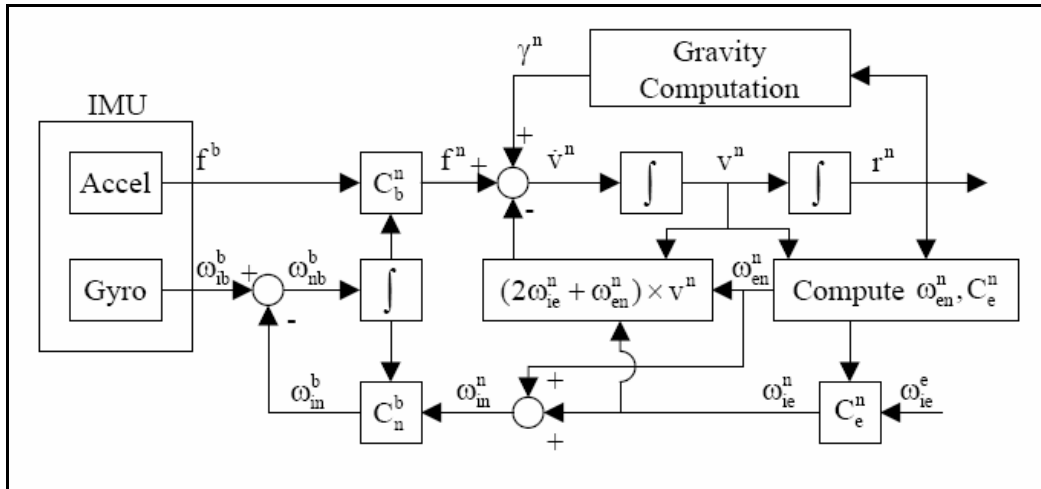


Figure 15: Block Diagram of INS mechanisation

### 5.4 Kalman Filter Module

The implementation of the Kalman filter was based on previous work by Kumar (2004) and Shin (2001). A 9 state filter, estimating position, velocity and attitude errors, was implemented in this research. This section will

present the method and equations that were used to implement the Kalman filter.

### 5.4.1 Error Dynamics

Perturbation methods are utilised in error analysis to linearise the nonlinear differential equations which define the error dynamics. The perturbation of position, velocity and attitude DCM can be expressed as:

$$\hat{r}^n = r^n + \delta r^n \quad (5.13)$$

$$\hat{v}^n = v^n + \delta v^n \quad (5.14)$$

$$\hat{C}_b^n = (I - E^n)C_b^n \quad (5.15)$$

where  $E^n$  is defined as the skew symmetric matrix of the attitude errors:

$$E^n = (\varepsilon^n \times) = \begin{bmatrix} 0 & -\varepsilon_D & \varepsilon_E \\ \varepsilon_D & 0 & -\varepsilon_N \\ -\varepsilon_E & \varepsilon_N & 0 \end{bmatrix} \quad (5.16)$$

By perturbing the dynamic equations of position, velocity and the attitude DCM (as defined in Chapter 4) the error dynamics equations can be derived. The error dynamics for position, velocity and attitude DCM are defined as:

$$\delta \mathcal{R} = F_{rr} \delta r^n + F_{rv} \delta v^n \quad (5.17)$$

$$\delta \mathcal{V} = F_{vr} \delta r^n + F_{vv} \delta v^n + (f^n \times) \varepsilon^n + C_b^n \delta f^b \quad (5.18)$$

$$\mathcal{E} = F_{er} \delta r^n + F_{ev} \delta v^n - (\omega_{in}^n \times) \varepsilon^n - C_b^n \delta \omega_{ib}^b \quad (5.19)$$

For a complete definition of these equations please refer to Appendix B.

Augmenting equations 5.17-5.19 produces a continuous time system defined as:

$$\dot{x} = Fx + Gu \quad (5.20)$$

where  $F$  is the dynamics matrix,  $x$  is the state vector,  $G$  is a design matrix and  $u$  is the forcing vector function (Shin, 2001):

$$F = \begin{bmatrix} F_{rr} & F_{rv} & 0 \\ F_{wr} & F_{wv} & (f^n \times) \\ F_{er} & F_{ev} & -(\omega_m^n \times) \end{bmatrix} \quad x = \begin{bmatrix} \mathcal{E}^n \\ \mathcal{V}^n \\ \mathcal{E}^n \end{bmatrix}$$

$$G = \begin{bmatrix} 0 & 0 \\ C_b^n & 0 \\ 0 & -C_b^m \end{bmatrix} \quad u = \begin{bmatrix} \mathcal{G}^b \\ \mathcal{E}_{ib}^b \end{bmatrix}$$

The elements of  $u$  are white noise with its covariance matrix defined as:

$$E[u(t)u(t)^T] = Q(t)\delta(t - \tau) \quad (5.21)$$

where  $\delta$  denotes the Dirac delta function whose unit is 1/time and  $Q$  is the spectral density matrix of the form (Shin, 2001):

$$Q = \text{diag}(\sigma_{ax}^2, \sigma_{ay}^2, \sigma_{az}^2, \sigma_{wx}^2, \sigma_{wy}^2, \sigma_{wz}^2) \quad (5.22)$$

and  $\sigma_a, \sigma_w$  are the standard deviations of the accelerometers and gyroscopes respectively.

Equation 5.20 is transformed to its discrete time form due to the high sample rate of strap-down INS. The discrete time form of equation 5.20 is defined as:

$$x_{k+1} = \Phi_k x_k + w_k \quad (5.23)$$

where  $\Phi_k$  is the state transition matrix and  $w_k$  is the random system process noise. The state transition matrix can be approximated using equation 5.24. This is a result of the small sampling time of the INS (Kumar, 2004). The

covariance matrix that is associated with system noise  $w_k$  is approximated as in equation 5.25.

$$\Phi_k \approx I + F\Delta t \quad (5.24)$$

$$Q_k \approx \Phi_k G Q G^T \Phi_k^T \Delta t \quad (5.25)$$

The norm of  $Q_k$  determines if the Kalman filter trusts the measurements more than the system. If the norm is larger than the actual norm the filter trusts the measurements, thus allowing free passage of measurement noise. Conversely, a norm that is much smaller than the actual norm will result in numerical instabilities in the filter. Hence, for a low cost INS,  $Q_k$  must be selected pessimistically so that the trajectory follows that of the GPS (Kumar, 2004), (Shin, 2001). Furthermore, the elements corresponding to altitude must account for both sensor imperfections and gravity.

The derivation of the Kalman filter is implemented using the random process model as presented in equations 4.13 and 4.14. The observation equation, 4.14, states that the measurement  $z_k$  is a combination of the state vector  $x_k$  and a random measurement error,  $v_k$ , with its covariance matrix defined as:

$$E(v_k v_k^T) = R_k \quad (5.26)$$

and  $R_k$  is defined as:

$$R_k = \text{diag}(\sigma_\lambda^2, \sigma_\mu^2, \sigma_h^2, \sigma_{vn}^2, \sigma_{ve}^2, \sigma_{vd}^2) \quad (5.27)$$

where  $\sigma_\lambda^2, \sigma_\mu^2$  and  $\sigma_h^2$  are the standard deviations of the GPS position and  $\sigma_{vn}^2, \sigma_{ve}^2$  and  $\sigma_{vd}^2$  are the standard deviations of the GPS velocities. These values were obtained from the GPS specifications. Alternatively, they can be obtained from GPS processing.

Using the equations 4.15-4.19, the Kalman filter can be implemented. Finally, the measurement vector,  $z_k$ , for the Kalman filter can be calculated as follows:

$$z_k = \begin{pmatrix} r_{INS}^n - r_{GPS}^n \\ v_{INS}^n - v_{GPS}^n \end{pmatrix} \quad H_k = \begin{pmatrix} I_{3 \times 3} & | & 0_{3 \times 3} & | & 0_{3 \times 3} \\ 0_{3 \times 3} & | & I_{3 \times 3} & | & 0_{3 \times 3} \end{pmatrix} \quad (5.28)$$

Unfortunately, this approach causes numerical instabilities when calculating the Kalman gain. This is due to the radian form of the latitude and longitude, as they are very small. To overcome this problem, the first two rows of  $H$  are multiplied by  $(M + h)$  and  $(N + h) \cos \lambda$ , respectively (Shin, 2001).

The initialisation of the Kalman filter is assumed to be performed when the vehicle is stationary. In such a case the position uncertainty will be equal to that of the GPS uncertainty, while the velocity uncertainty is equal to zero. The attitude uncertainty depends on the biases of the accelerometers and gyroscopes. The attitude uncertainty can be improved if the biases are estimated properly (Shin, 2001).

The Kalman filter state estimate,  $x_k$ , is the Kalman filter output. In this case, the output is the estimated position, velocity and attitude errors. Therefore, it is  $x_k$  that is used for INS correction. There exist two methods for INS correction, feed-forward and feed-back (as discussed in section 4.5.1). With a feed-forward mechanisation the INS errors continue to grow with time as they are never corrected within the INS. As a result, unbounded error observations are provided to the Kalman filter, which in turn causes problems to the linear Kalman filter implemented. Therefore, a feed-back correction mechanism has been implemented. The feed-back equations are given below:

$$r^n = \hat{r}^n - \delta r^n \quad (5.29)$$

$$v^n = \hat{v}^n - \delta v^n \quad (5.30)$$

$$C_b^n = (I + E^n) \hat{C}_b^n \quad (5.31)$$

After feed-back is performed, the state vector  $x_k$  is reset to zero as the filter assumes zero error until a measurement is made.

## ***6.0 Results & Discussion***

This Chapter presents the results obtained for the different components implemented. Each component is analysed separately. The OHBY mission planning system is discussed. This is followed by the path planning testing. Finally, the navigation results are discussed.

### ***6.1 OHBY Mission Planning***

A test plan was developed based on the features of OHBY. This test plan describes the approach to testing and validating the operation and effectiveness of OHBY. Specifically, the test plan describes the required resources, features to be tested, the testing methodology and the test deliverables. A test case for each function has been specified. The test case consists of example input and the expected output for each function. This approach to testing is useful as it can provide insights into possible code errors within OHBY. The test plan was put into practice upon the completion of the implementation phase. The test plan revealed that the OHBY features function as expected, as all the test cases were passed successfully. For more detail please refer to Appendix C.

### ***6.2 ELOS Path Planning***

The testing of the path planning was performed in 3 stages. Stage 1 was the testing of the ELOS 2D algorithm. The ELOS 3D algorithm was tested in stage 2, while the further extensions to the ELOS 3D algorithm were tested in stage 3.

### 6.2.1 ELOS 2D Algorithm

The ELOS 2D algorithm has been incorporated into the OHBY mission planner. Therefore, the testing of the 2D path planner was performed as part of the OHBY test plan.

The testing involved the input of two or more way-points, as well as a numerous number of objects into the map area. The path planner was then initiated using the *Solve Path* command in the *Edit Mission* menu. The purpose of testing the application was to validate the operation of the algorithm. Such testing was critical to ensure that the 2D algorithm was operational, so that the 3D path planning extension could be implemented.

Figures 16 and 17 show the generated paths based on the objects present. It can be seen that the algorithm is capable of determining an optimal path even with a large number of obstacles present.

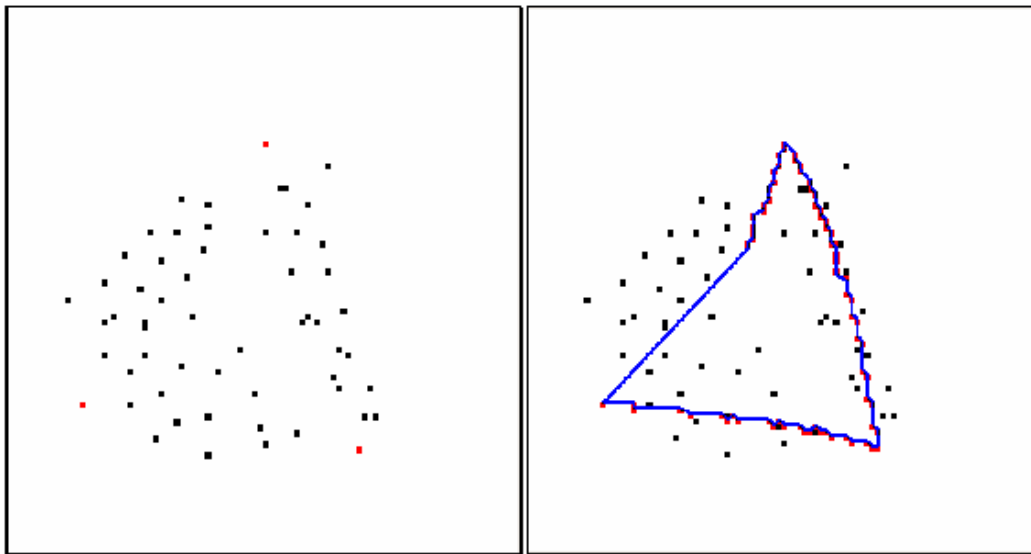


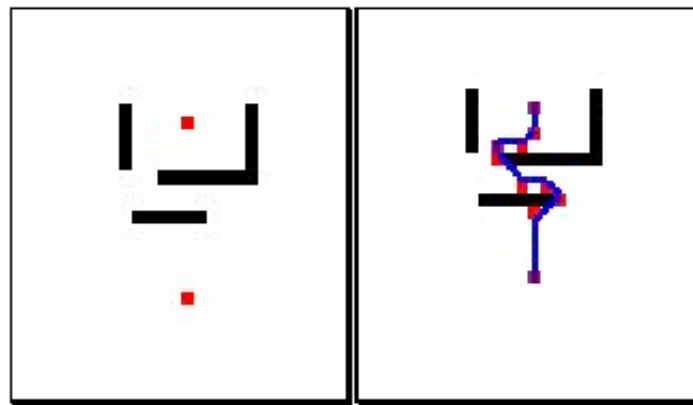
Figure 16: ELOS 2D Path Planning: Test 1

Test 1 required the UAV to fly a triangular course between 4 way-points. The static objects that were input were numerous and scattered. To plot the path,

the path planner traverses the list and for each way-point pair, it generates an optimal path. The way-point pairs for this test case were:

- Way-Point 0 – Way-Point 1
- Way-Point 1 – Way-Point 2
- Way-Point 2 – Way-Point 3

The generated path, shown in the right half of Figure 16, has input numerous way-points into the mission list to guide the UAV through the obstacles. The inserted way-points represent turns that the UAV will need to perform to avoid the obstacles.



**Figure 17: ELOS 2D Path Planning, Test 2**

In test 2, as shown in Figure 17, the input objects were designed to create an urban environment. Each block of objects can represent a building or a block of houses. It can be seen that the path planner generates an optimal path through the obstacles.

During the testing of the path planner, it was found that in some circumstances the path planner does not generate an optimal path. One such example is demonstrated in Figure 18 below.

1)	<b>x,y</b>	<b>50</b>	.....	<b>60</b>	<b>61</b>	<b>62</b>	.....	<b>98</b>
	<b>49</b>		.....	48.06303	48.05887		.....	
	<b>50</b>	Target	.....	Object	Current		.....	Start

2)	<b>x,y</b>	<b>50</b>	.....	<b>60</b>	<b>61</b>	<b>62</b>	.....	<b>98</b>
	<b>49</b>		.....	48.06303	Current	48.05548	.....	
	<b>50</b>	Target	.....	Object			.....	Start

3)	<b>x,y</b>	<b>50</b>	.....	<b>60</b>	<b>61</b>	<b>62</b>	.....	<b>98</b>
	<b>49</b>		.....			Current	.....	
	<b>50</b>	Target	.....	Object			.....	Start

**Figure 18: Time snapshots of the 2D array**

Figure 18 presents three time varied arrays. Array 1, displays the detection of the object. At this point the path planner calculates the shortest distance through the obstacle free cells. However, using the equilibrium distance formula, equation 5.1, yields the shortest distance as cell (49, 61) instead of cell (49, 60). In the next step (array 2), the shortest distance is determined to be cell (49, 62). Therefore, the generated path is now heading back towards the start rather than to the target.

This problem can generally be overcome by running the path planner twice, as is currently the case. However, to ensure correct operation at first run, an adjustment to the distance calculation was made. To ensure that the path is optimal relative to the target, the distance from current location to target is given more weight than the distance from current location to start. Therefore, the new distance equation is defined as:

$$dist = 0.5\sqrt{(x_s - x_c)^2 + (y_s - y_c)^2} + \sqrt{(x_t - x_c)^2 + (y_t - y_c)^2} \quad (6.1)$$

Similarly, the new equation for the 3D path planner is defined as:

$$dist = 0.5\sqrt{(x_s - x_c)^2 + (y_s - y_c)^2 + (z_s - z_c)^2} + \sqrt{(x_t - x_c)^2 + (y_t - y_c)^2 + (z_t - z_c)^2} \quad (6.2)$$

This weighting scheme was found to overcome the aforementioned problem.

### 6.2.2 ELOS 3D Algorithm

Due to time constraints, the 3D extension to the ELOS algorithm could not be integrated with the OHBY mission planner. Therefore, a different approach to testing was applied. Instead of using a graphical mission planner, a text mission planner with minimal functionality was created. Using the text mission planner, the user was able to perform the same way-points operations as in OHBY. However, in the initial testing of the 3D algorithm the objects were hard-coded into the program.

Figure 19, shows the output of the 3D path planner. The green line represents the obstacles, which may be a tower or skyscraper; while, the red line is the optimal path generated.

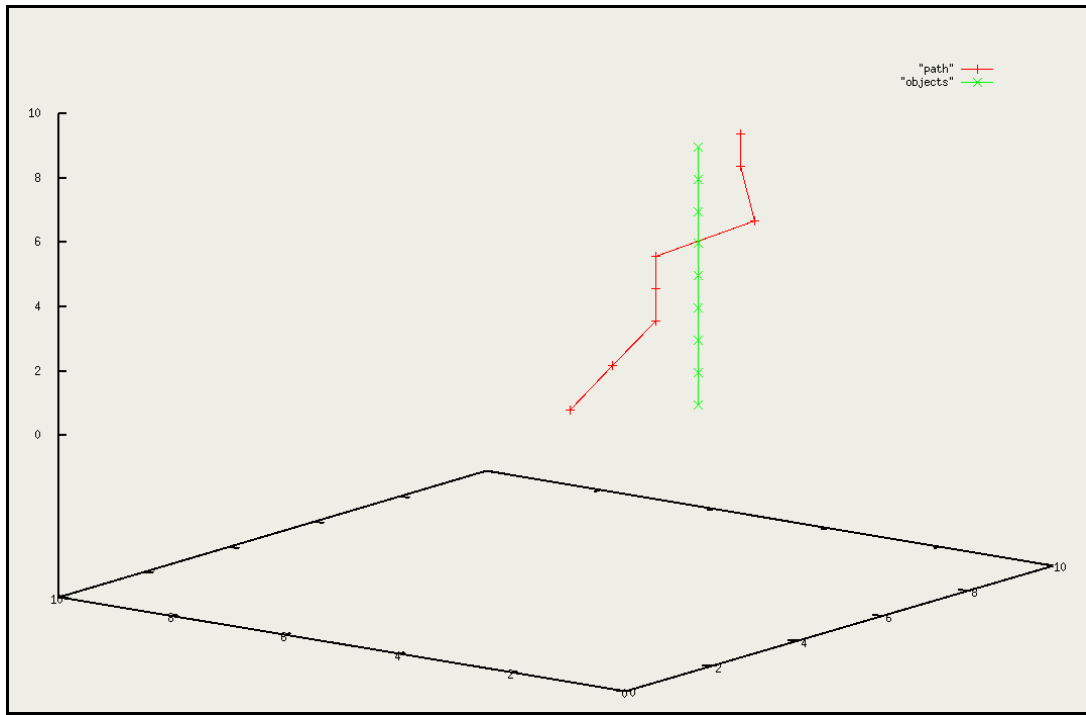


Figure 19: 3D path planner

The start and target locations, for Figure 19, are at the following x, y and z coordinates:

Start:           x=4   y=4   z=1

Target:          x=8   y=4   z=8

At first detection of the obstacle, the path generated requires the UAV to fly directly upwards. At  $z=5$ , the path leads the UAV around the obstacle and upwards to the target location. This example demonstrates that the 3D extension operates as required. Thus, the UAVs flight capabilities may be taken full advantage of.

Due to time constraints and the desire to implement a no-fly zone extension, extensive testing of the 3D path planner was not performed. However, experience with the 2D planner and initial test outcomes suggest that the 3D planner is fully functional.

### **6.2.3 No-Fly Zones Extension**

The no-fly zones extension applied to the ELOS 3D algorithm was tested in stage 3. The text mission planner was utilised again, and in addition to obstacles, a no-fly zone was defined as specified in section 5.2.3.

Figure 20 below, shows a side view of the generated path. The blue lines represent the no-fly zone that has been defined, while the red and green lines representing the path and objects, respectively. Figure 20 clearly shows that the path does not enter the no-fly zone. To verify that this is indeed a result of the extension, the generated path was compared with Figure 19. The comparison identified that without a specified no-fly zone, the UAV would fly directly through that airspace.

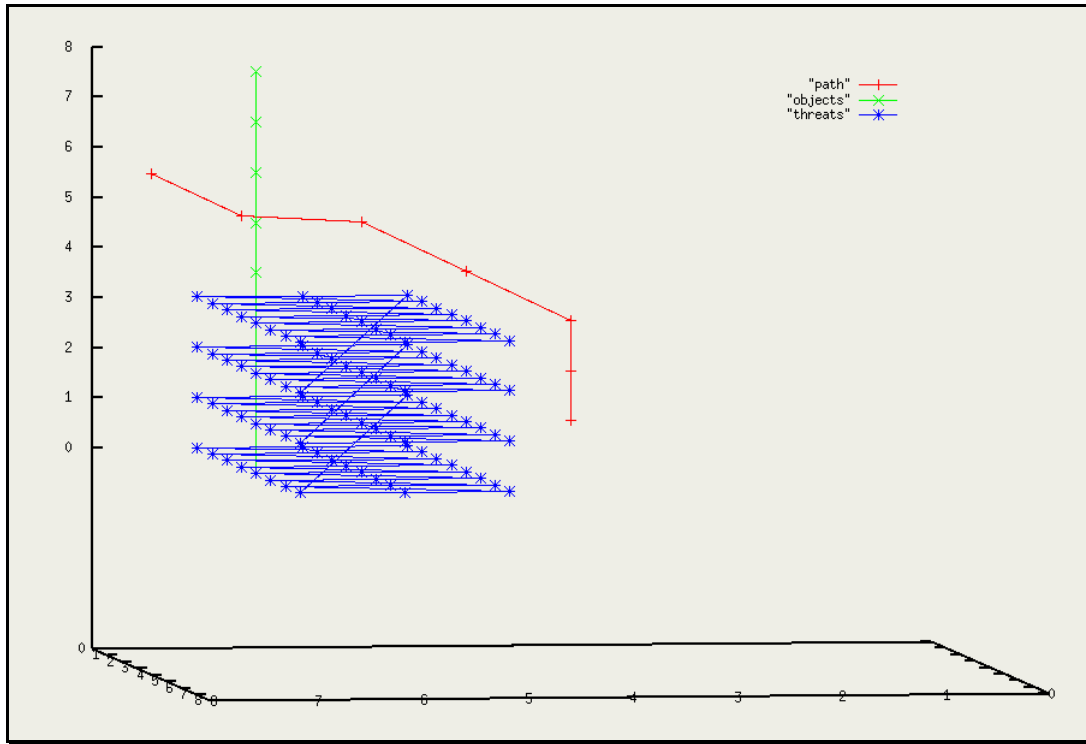


Figure 20: No-fly Zone path generation, Side view

Figure 21 shows the top view of the generated path. This view also clearly demonstrates that the UAV is directed to avoid the defined object.

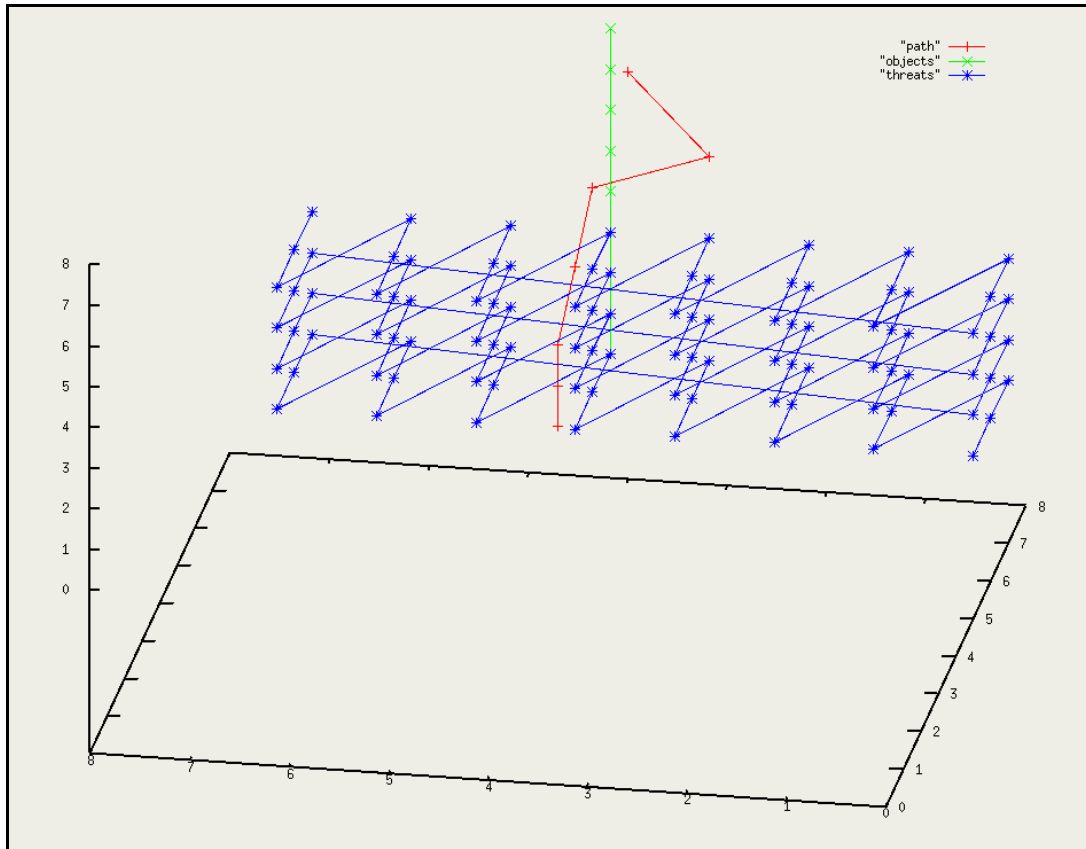


Figure 21: No-fly Zone path generation, Top view

It must also be noted that the 3D paths generated in some instances require the UAV to fly directly up. For fixed wing UAVs this would be unfeasible; however, for a rotary UAV, which is the focus of this project, such manoeuvres are not a problem.

### 6.3 Navigation

Three individual components of navigation were tested in order to determine whether the navigation system was fully functional and if any limitations were present to assist in its easier identification. Thus, the sensor module required testing to confirm correct operation. The INS was tested using collected IMU data. Finally, the INS was integrated with the Kalman filter to complete the navigation system. This section discusses the results and findings of these experiments.

### 6.3.1 IMU

The software modules developed to read the IMU data were tested to confirm the axes and angle directions. It was found that the axes were in fact opposite to the directions required. Furthermore, during the validation of IMU readings in ASCII mode, it was determined that the  $x$  and  $y$  gyroscope values were not sent in the expected format. From the testing it was concluded that:

- $x$ ,  $y$  and  $z$  accelerometer readings must be flipped
- $x$  and  $y$  gyroscope values need to be swapped
- $x$  and  $y$  gyroscope values need to be flipped

By performing these corrections, in the INS, the body frame will be aligned with the navigation frame of the form North, East and Down. Furthermore, the IMU testing in binary mode found that in the binary format the  $x$  and  $y$  gyroscope values need not be swapped.

### 6.3.2 INS

The INS could not be tested without IMU data. IMU data was collected by mounting the IMU onto a car. The IMU was connected to a laptop and using the created software the data was collected and stored into files for future reuse. The data files contain accelerometer, gyroscope and GPS values that were gathered whilst driving around the Monash University, Clayton campus. To test the INS these data files were read and adjusted, as identified in the previous section. Subsequently the INS process was run.

Figure 22 shows the latitude and longitude output of the INS. The green dots in the diagram are the GPS readings gathered during the drive. The red dots are the INS output.

It is clear that the INS output has severe errors. This was expected as the IMU that was used is of low-cost, and therefore has substantial errors. To improve the INS a degree of error modelling needs to be implemented. Therefore, bias

and scale correction was implemented in code. However, determining the actual bias and scale values is beyond the scope of this research. Nevertheless, the equations for the correction have been implemented in code; the equations that were used are supplied in Appendix A.

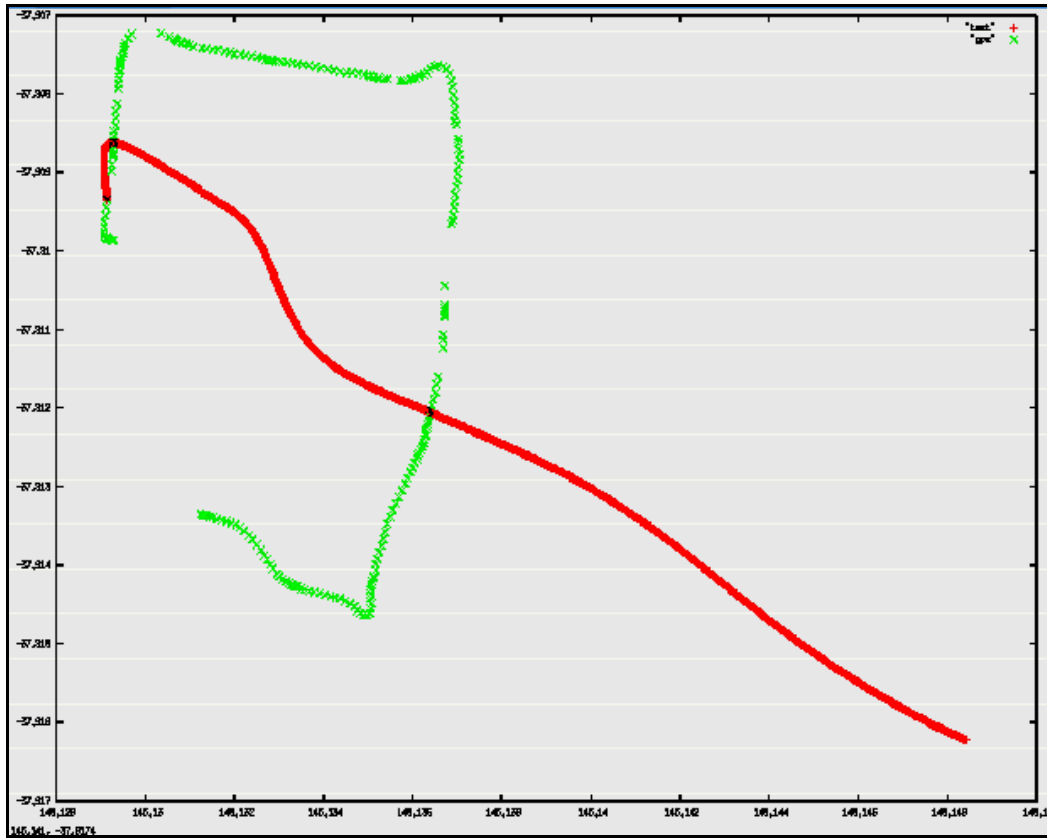


Figure 22: INS Latitude and Longitude

Furthermore, it is noticeable that in several locations a GPS outage occurs. This serves to highlight the need for an INS/GPS integrated navigation system, designed for a rotary UAV.

### 6.3.3 Kalman Filter

The Kalman filter testing was performed using the data files gathered for INS testing. The Kalman filter was used to integrate the GPS values gathered with the INS calculations.

Figure 23, below, displays the results of the Kalman filtering. As before, the green dots represent the GPS readings and the red dots are the INS output.



Figure 23: Kalman Filter results

As displayed in Figure 23, it is evident that the Kalman filter does not perform the error correction as expected. By comparing the GPS values from Figure 22, it can be seen that the INS output of Figure 23 follows the GPS path.

However, the distances involved with the INS outputs are highly exaggerated. Furthermore, the INS outputs are eventually corrupted by the Kalman filter. This is a result of the Kalman filter diverging as the error observations grow without bound. Unfortunately, due to time constraints the error source could not be identified.

## ***7.0 Conclusion & Future Work***

In this thesis several crucial components of an UAV system have been designed and implemented. Namely, these components are the mission and path planning, as well as the navigation system. The OHBY mission planner was developed as a framework for future development for a ground station. Currently, OHBY enables mission planning via user specified way-points. Furthermore, a path planning algorithm was implemented and extended upon to enable the UAV to avoid obstacles. These components provide the baseline requirements of a mission and path planner. Testing of the components via simulations confirms their functionality. Unfortunately, the implementation of the navigation system was not as successful. Test results suggest the INS is operational; however, the integration of GPS via the Kalman filter is not as effective as expected. These limitations could not be rectified due to time constraints. Nevertheless, the components in their current state provide a great deal of functionality and a beneficial framework for potential future work.

This chapter discusses some of the limitations of the research undertaken and potential future work is recommended. The Kalman filter and the evolution of the path and mission planners are of most concern.

### **7.1 Limitations**

The most significant limitation in this project is the error present in the Kalman filter. Although the results are promising; an implementation or design error has caused the filter to diverge. This limitation is of utmost importance as navigation for an UAV is critical. Therefore, identifying and rectifying the error source is crucial for the further development of the UAV.

## 7.2 Future Work

Since this is an evolving project there exist many possible paths for future work. Based on the framework provide by this thesis, it would be most beneficial if work on these components were continued.

Obviously rectifying the error present in the Kalman filter is essential to provide navigation for the UAV. Furthermore, error modelling of the sensor biases and scale factors need to be performed in order to improve the accuracy of the INS. By performing these tasks a reliable and accurate navigation system would be available for the UAV.

The further developments of the mission and path planners are beneficial. The integration of the OHBY mission planning system with the 3D path planner would be an obvious start. This might require the development of a graphical 3D display of the flight path generated. In addition, the mission planner can be expanded so that fuel and flight time are considered when planning a mission.

Further extensions of the path planner should also be considered. Currently, a no-fly zone extension has been implemented; however, assigning threat levels to each cell, based on provided input and a probability function, may be considered.

## 8.0 References

- Akram, M.I. Pasha, A. and Iqbal, N. 2004, 'Optimal Path Planner for Autonomous Vehicles', Transactions on Engineering, Computing and Technology
- Bortoff, S. A. 2000, 'Path planning for UAVs', In *Proc. of the American Control Conference*, pages 364–368, Chicago, IL.
- Cramer, M. 1997, 'GPS/INS Integration', University of Stuttgart
- Dixon, J. and Henlich, O. 1997. Mobile Robot Navigation – Final Report. Imperial College, London, 1997.  
Retrieved 4 June 2006, from  
[http://www.doc.ic.ac.uk/~nd/surprise\\_97/journal/vol4/jmd](http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/jmd)
- Dollery, W. 2001. Autonomous Flying Robots (Helicopter) Development of Processing Platform and System Design. Honours, Monash University, Clayton, Victoria, Australia.
- Farrell, J.A., and Barth, M. 1998. *The Global Positioning System & Inertial Navigation*. McGraw-Hill.
- Grewal, M. S., Weill, L. R., and Andrews, A. P. 2001. *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley & Sons, Inc.
- Hall, M. R. 1988, 'A Mission Planning Architecture for An Autonomous Vehicle', Artificial Intelligence Laboratory
- Hide, C Moore, T. 2005, 'GPS and Low Cost INS Integration for Positioning in the Urban Environment', University of Nottingham
- Jun, M. D'Andrea, R. 2000, 'Path Planning for Unmanned Aerial Vehicles In Uncertain and Adversarial Environments', in the Book "Cooperative Control: Models, Applications and Algorithms"

- Krishnamurthy Gopalan, A. Davari, A. Manish, A. , 'Optimal Path Planning for an Unmanned Aerial Vehicle', WVU Tech
- Kumar, V., 2004, 'Integration of Inertial Navigation System and Global Position System Using Kalman Filtering' Department of Aerospace Engineering Indian Institute of Technology, Mumbai.
- Lee, J. Huang, R. Vaughn, A. Xiao, X. and Karl Hedrick, J. 2003 , 'Strategies of Path-Planning for a UAV to Track a Ground Vehicle', University of California
- Levy, J. 1997, *The Kalman Filter: navigation's Integration Workhorse*, GPS World, September, pp 65 - 71.
- Li, Y., Wang, J., Rizos, C. Mumford, P., Ding, W. 2005, 'Low-cost tightly coupled GPS/INS integration on a nonlinear Kalman filtering design', University of New South Wales
- Minkler G., Minkler J. 1993, *Theory and Application of Kalman Filtering*, Magellan Book Company
- Nelson, M.J. 1995, 'UAV Mission Planning', DSTO Report
- Newcome, L.R. 2004. *Unmanned Aviation: A brief history of unmanned aerial vehicles*. Reston, Virginia: American Institute of Aeronautics and Astronautics.
- O-Navi, (2006), Retrieved on 5<sup>th</sup> November 2006, Retrieved from <http://www.o-navi.com/products.htm>
- Quach, N., Harvey, L. and Sinclair, R. 2004. *The Design of an Autonomous Small Scale Helicopter*. Monash University, Clayton, Victoria, Australia
- Saggiani, G.M., Teodorani, B. 2003, 'Rotary Wing UAV Potential Applications: An Analytical Study through a Matrix Method', University of Bologna, Bologna, Italy.

Shin, E.H. 2001, 'Accuracy Improvement of Low Cost INS/GPS for Land Application,' University of Calgary

Titterton, D. H. and Weston, J. L. 1997. *Strapdown Inertial Navigation Technology*. Peter Peregrinus Ltd.

Walchko, K. Mason, P. 2002,'Inertial Navigation', Florida Conference on Recent Advances in Robotics

Watkinson, J. 2004. *The Art of the Helicopter*. Oxford: Butterworth-Heinemann.

Wolf, R. Eissfeller, B. Gunter W.H. 1997,'A Kalman Filter for the Integration of sa low cost INS and an attitude GPS', Institute of Geodesy and Navigation, Berlin, Germany.

Zelek, J.S. 1995, 'Dynamic path planning'. In IEEE Conference on Systems, Man and Cybernetics

# Appendix A

## Inertial Navigation Equations

### *Direction Cosine Matrices*

- Earth to Navigation frame

$$C_e^n = \begin{bmatrix} -\sin \lambda \cos \mu & -\sin \lambda \sin \mu & \cos \lambda \\ -\sin \mu & \cos \mu & 0 \\ -\cos \lambda \cos \mu & -\cos \lambda \sin \mu & -\sin \lambda \end{bmatrix} \quad (\text{A.1})$$

- Navigation to Earth frame

$$C_n^e = (C_e^n)^T = \begin{bmatrix} -\sin \lambda \cos \mu & -\sin \mu & -\cos \lambda \cos \mu \\ -\sin \lambda \sin \mu & \cos \mu & -\cos \lambda \sin \mu \\ \cos \lambda & 0 & -\sin \lambda \end{bmatrix} \quad (\text{A.2})$$

- Body to Navigation frame

$$C_b^n = (C_n^b)^T = \begin{bmatrix} \cos \theta \cos \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \quad (\text{A.3})$$

### *Position and Velocity Equations*

- Acceleration is defined as

$$\mathfrak{A} = f^n - (2\omega_{ie}^n + \omega_{en}^n) \times V_e^n + g_l^n \quad (\text{A.4})$$

where

$$\omega_{ie}^n = (\omega_e \cos \lambda, 0, -\omega_e \sin \lambda)^T \quad (\text{A.5})$$

$$\omega_{en}^n = (v_E / (N+h), -v_N / (M+h), -v_E \tan \lambda / (N+h))^T \quad (\text{A.6})$$

and  $\omega_e$  is the earth rotation rate and is defined as 7.2921158 rad/s.

M, N are the meridional radius of curvature and transverse radius of curvature and are defined as:

$$\begin{aligned} N &= \frac{a}{(1-e^2 \sin^2 \lambda)^{1/2}} \\ M &= \frac{a(1-e^2)}{(1-e^2 \sin^2 \lambda)^{3/2}} \end{aligned} \quad (\text{A.7})$$

where  $a$  and  $e$  are the semi-major axis and linear eccentricity of the ellipsoid, respectively.

$g_i^n$  is the term combining acceleration due to gravity and the centripetal acceleration. It is defined as:

$$g_i^n = g^n - \left( \frac{\omega_e^2 (R_e + h)}{2} \right) [\sin(2\lambda), 0, 1 + \cos(2\lambda)]^T \quad (\text{A.8})$$

where the acceleration due to gravity  $g^n = [0, 0, g]^T$  and  $g$  is gravity at sea-level.

### ***Attitude Equations***

The propagation of  $C_b^n$  can be defined as:

$$\mathcal{C}_b^n = C_b^n (\omega_{nb}^b \times) = C_b^n ((\omega_{ib}^b - \omega_{in}^b) \times) \quad (\text{A.9})$$

where  $\omega_{ib}^b$  is the vector of gyroscope readings from the IMU and  $\omega_{in}^b$  is the sum of the earth rotation rate and the transport rate,  $\omega_{in}^n$ , projected onto the body frame. Therefore,  $\omega_{in}^b$  can be defined as:

$$\omega_{in}^b = C_n^b \omega_{in}^n \quad (\text{A.10})$$

### ***Error Compensation***

The corrected gyroscope and accelerometer readings are defined as:

$$\Delta \omega_{ib}^b = \Delta \tilde{\omega}_{ib}^b - b_\omega \Delta t \quad (\text{A.11})$$

$$\Delta v_f = \begin{bmatrix} 1/(\mathbf{1} + s_{gx}) & 0 & 0 \\ 0 & 1/(\mathbf{1} + s_{gy}) & 0 \\ 0 & 0 & 1/(\mathbf{1} + s_{gz}) \end{bmatrix} (\Delta \tilde{v}_f - b_g \Delta t) \quad (\text{A.12})$$

where  $b_\omega$  and  $b_g$  are the gyroscope and accelerometer biases, respectively.  $s_{gx}, s_{gy}$  and  $s_{gz}$  are the accelerometer scale factors

## Appendix B

### Error Dynamics

#### *Position Error Dynamics*

The error dynamics for position is defined as:

$$\delta\mathcal{E} = F_{rr}\delta r^n + F_{rv}\delta v^n \quad (\text{B.1})$$

where

$$F_{rr} = \begin{pmatrix} \frac{\partial\lambda}{\partial\lambda} & \frac{\partial\lambda}{\partial\mu} & \frac{\partial\lambda}{\partial h} \\ \frac{\partial\dot{\mu}}{\partial\lambda} & \frac{\partial\dot{\mu}}{\partial\mu} & \frac{\partial\dot{\mu}}{\partial h} \\ \frac{\partial\dot{h}}{\partial\lambda} & \frac{\partial\dot{h}}{\partial\mu} & \frac{\partial\dot{h}}{\partial h} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{-V_N}{(R_e+h)^2} \\ \frac{V_E \sin\lambda}{(R_e+h)\cos^2\lambda} & 0 & \frac{-V_E}{(R_e+h)^2\cos\lambda} \\ 0 & 0 & 0 \end{pmatrix}$$

$$F_{rv} = \begin{pmatrix} \frac{\partial\lambda}{\partial V_N} & \frac{\partial\lambda}{\partial V_E} & \frac{\partial\lambda}{\partial V_D} \\ \frac{\partial\dot{\mu}}{\partial V_N} & \frac{\partial\dot{\mu}}{\partial V_E} & \frac{\partial\dot{\mu}}{\partial V_D} \\ \frac{\partial\dot{h}}{\partial V_N} & \frac{\partial\dot{h}}{\partial V_E} & \frac{\partial\dot{h}}{\partial V_D} \end{pmatrix} = \begin{pmatrix} \frac{1}{R_e+h} & 0 & 0 \\ 0 & \frac{1}{(R_e+h)\cos\lambda} & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

and  $R_e$  is the radius of the earth and is considered a constant.

#### *Velocity Error Dynamics*

The error dynamics for velocity are defined as:

$$\delta\mathcal{E} = F_{vr}\delta r^n + F_{vv}\delta v^n + (f^n \times)\varepsilon^n + C_b^n \delta f^b \quad (\text{B.2})$$

where

$$F_{vr} = \left( \begin{array}{c|c|c} \frac{-2V_E\Omega \cos \lambda - \frac{V_E^2}{(R_e+h)\cos^2 \lambda}}{2\Omega(V_N \cos \lambda - V_D \sin \lambda) + \frac{V_E V_N}{(R_e+h)\cos^2 \lambda}} & 0 & \frac{-V_N V_D}{(R_e+h)^2} + \frac{V_E^2 \tan \lambda}{(R_e+h)^2} \\ \hline & 0 & \frac{V_E V_D}{(R_e+h)^2} - \frac{V_N V_E \tan \lambda}{(R_e+h)^2} \\ \hline 2V_E\Omega \sin \lambda & 0 & \frac{V_E^2 + V_N^2}{(R_e+h)^2} - \frac{2\gamma}{(R_e+h)} \end{array} \right)$$

$$F_{vv} = \left( \begin{array}{c|c|c} \frac{V_D}{R_e+h} & -2\Omega \sin \lambda - 2\frac{V_E \tan \lambda}{R_e+h} & \frac{V_N}{R_e+h} \\ \hline \frac{2\Omega \sin \lambda + \frac{V_E \tan \lambda}{R_e+h}}{-2\frac{V_N}{R_e+h}} & \frac{V_D + V_N \tan \lambda}{R_e+h} & 2\Omega \cos \lambda + \frac{V_E}{R_e+h} \\ \hline & -2\Omega \cos \lambda - 2\frac{V_E}{R_e+h} & 0 \end{array} \right)$$

and  $\Omega$  is the earth rotation rate.

### ***Attitude Error Dynamics***

The attitude error dynamics are defined as:

$$\dot{\mathcal{E}} = F_{er} \delta r^n + F_{ev} \delta v^n - (\omega_{in}^n \times) \mathcal{E}^n - C_b^n \delta \omega_{ib}^b \quad (\text{B.3})$$

where

$$F_{er} = \left( \begin{array}{ccc|c} -\Omega \sin \lambda & 0 & \frac{-V_E}{(R_e+h)^2} \\ 0 & 0 & \frac{V_N}{(R_e+h)^2} \\ -\Omega \cos \lambda - \frac{V_E}{(R_e+h)\cos^2 \lambda} & 0 & \frac{V_E \tan \lambda}{(R_e+h)^2} \end{array} \right)$$

$$F_{ev} = \left( \begin{array}{ccc|c} 0 & \frac{1}{R_e+h} & 0 \\ \frac{-1}{R_e+h} & 0 & 0 \\ 0 & \frac{-\tan \lambda}{R_e+h} & 0 \end{array} \right)$$

# *Appendix C*

## OHBY Mission Planning System Test Plan

### **1. Introduction**

#### **Description of this Document**

This document is a Test Plan for the OHBY Mission Planning System. The document describes the testing strategy and procedures employed to validate effective and efficient operation of the product. OHBY focuses on providing assistance for planning a UAV mission.

The features that will be tested include:

- Way-Point input
- Object input
- Boundary translation
- Path planner
- Mission commands

#### **Schedule and Milestones**

Testing should take approximately 20 minutes.

### **2. Resource Requirements**

#### **Software**

- Glade-2

### **3. Features to Be Tested / Test Approach**

- Way-Point input
- Object input
- Boundary translation
- Path planner
- Mission commands

#### **Way-Point input**

Way-points will be added, inserted, modified and deleted to the mission to determine correct operation

#### **Object input**

Objects will be input interactively and manually to validate the feature

#### **Boundary Translation**

Way-points and objects will be added to the main screen, a boundary translation is performed to determine correct translation of the boundaries

#### **Path Planner**

Way-points and objects are entered and the path planner is initiated to solve the path. This will determine correct operation of the algorithm

#### **Mission Commands**

A mission will be created and the mission commands executed.

### **4. Features Not To Be Tested**

Manual controls

## 5. Test Deliverables

### Way-Point Input

Way-point addition

Test case	Expected outcome	Pass/Fail
Add way-point 1	Way-point added	Pass
Add way-point 2	Way-point added	Pass

Way-point insertion

Test case	Expected outcome	Pass/Fail
Insert way-point 3	Way-point inserted	Pass
Insert way-point 4	Way-point inserted	Pass

Way-point deletion and modification

Test case	Expected outcome	Pass/Fail
Remove way-point 4	Way-point 4 removed	Pass
Modify way-point 3	Way-point 3 modified	Pass

NB. A Way-point can be a series of any latitude, longitude and altitude coordinates.

### Object Input

Object addition and removal

Test case	Expected outcome	Pass/Fail
Add Object 1	Object added	Pass
Remove Object 1	Object removed	Pass

Interactive object addition and removal

Test case	Expected outcome	Pass/Fail
Add Object 1	Object added	Pass
Remove Object 1	Object removed	Pass

NB. An object can be a series of coordinates representing latitude and longitude.

## Boundary Translation

Are the boundaries translated accordingly?

Test case	Expected outcome	Pass/Fail
Latitude translation 500m	Latitude translated	Pass
Longitude translation 500m	Longitude translated	Pass

## Path planner

Is path planner operational?

Test case	Expected outcome	Pass/Fail
2 way-point mission	Path solved	Pass
4 way-point mission	Path solved	Pass
n way-point mission	Path Solved	Pass

## Mission Commands

Are mission commands operational?

Test case	Expected outcome	Pass/Fail
Start Mission	Mission saved	Pass
Stop Mission		NA
Cont. Mission		NA
Cancel Mission		NA