NTNU – Trondheim
Norwegian University of
Science and Technology

POSITION CONTROL FOR AUTOMATIC LANDING OF UAV IN A NET ON SHIP

Kjetil Hope Sørbø

Submission date: May 2016
Responsible professor: Thor Inge Fossen, Tor Arne Johansen

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Contents

Acronyms

DGPS Differential GPS.

GLONASS Global Navigation Satellite System.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

IMC Inter-Module Communication.

INS Inertial Navigation System.

LAMBDA Least-squares AMBiguity Decorrelation Adjustment.

RTK-GPS Real Time Kinematic GPS.

RTKLIB Real-Time Kinematic Library.

SIL Software In the Loop.

TOW Time Of Week.

UAV Unmanned Aerial Vehicle.

WGS-84 World Geodetic System, 1984.

Chapter 1

Introduction

1.1 Background

Recent development of flying Unmanned Aerial Vehicles (UAVs) has been recognized to provide an attractive alternative to work previously performed by manned operations. Typical work which has attracted attention includes inspection, aerial photography, environmental surveillance and search and rescue. Today UAVs are mostly operated over land, however in the future this will include over sea as well. This will give some challenges which must be overcome. One of these challenges is that the UAV need to be able to perform a autonomous landing.

An UAV can provide an attractive alternative for many maritime operation where today manned aircraft or satellites is the only solution. In the maritime sector UAV can be used in iceberg management, monitoring of oil spills, search and rescue and maritime traffic monitoring.

An important premise for successful and safe UAV operation, in particular at sea, is the provision of a robust system for safe landing of the UAV on a vessel following completed operations. A autonomous landing system require a plan generation system that can create a flyable landing path during flight operation from any initial position that is within the operation criteria set by the operator. In addition the navigation system must have centimeter level accuracy in order for the UAV to perform a autonomous landing in a net. However with a accurate navigation system the case of what to do when the positioning system degenerates must be resolved such that system failure does not occur.

Due to regulatory mandate there are restriction on the size of operational area for a UAV. This thesis will only review a Line Of Sight (LOS) operation where the UAV must within view of the pilot during the duration of flight, which restrict the

area were a autonomous landing can take place.

1.2 Literature review

There exits today autoland system for fixed wing UAV that apply INS/GPS[Insitu], however this system require expensive equipment and is limited to a few UAV systems. The limitation on type of UAV and high cost restricts the usage of the recovery system, and motivates the research of a low cost recovery system for fixed wing UAV.

A low cost recovery system for fixed wing UAV is proposed in the paper [Kim et al., 2013], where computer vision is used to find and identify the recovery net. The system was successful in performing a autonomous landing, however it require that the visual image is sent from the UAV to a ground station. In addition the system require a clear image in order to calculate guidance command for the UAV, which restricts when the system can used.

1.3 Previous work

A low-cost net recovery system for UAV with single-frequency Real Time Kinematic GPS (RTK-GPS) was described in the paper [Skulstad et al., 2015], which was a result of the work done in the master thesis [Skulstad and Syversen, 2014]. The system presented applied RTKLIB together with low-cost single frequency Global Positioning System (GPS) receivers as navigation system with a customized Ardupilot software. The complete system was able to perform a net landing, however the result showed that further work would require better controllers, and a more robust navigation system.

A continuation of the work done in [Skulstad and Syversen, 2014] was done in [Frølich, 2015]. The work simulated a autonomous net landing, however no physical experiment was perform. The result in the work indicated that further work on the controllers was required, in addition to the landing path which was not suited for a Visual Line Of Sight (VLOS) UAV operation due to no spacial restrictions.

1.4 Contributions

This thesis focus on the navigation system and generation of landing path in the autonomous landing system. The navigation system apply RTK-GPS to provide high accuracy position estimation, which is needed to perform a autonomous landing. The landing path provide a flyable path from any initial position, where the length and direction of the virtual runway is determined by the operator. Through this work, the following contributions has been made:

- A landing path generator has been created, which guaranty a flyable path with controlled decent from any initial position 4.1.
- The landing path has been implemented in the DUNE runtime environment, which is capable to be used in both a stationary and moving net landing.
- A new Inter-Module Communication (IMC) message has been created to contain the landing path specifications.
- A net nest has been constructed to provide the GPS coordinates for the stationary net.
- A navigation state machine, which is used to control which position solution source should be used in the payload computer. The state machine will try to keep the RTK-GPS available as long as possible by adding the average difference to the position solution from the Pixhawk for a short duration of time until a new viable RTK-GPS message is received.
- A navigation source interface has been created to provide a visual indicator of which navigation system that is used in the DUNE environment.
- Physical experiments of the navigation system and landing path generator.

Chapter 2

Basis and modelling

2.1 UAV model

A UAV model can be described in different coordinate frames depending on which forces and moments are the focus area. The kinematics and kinetics equations (2.1)-(2.3) used to describe a MAV aircraft is found in [Beard and McLain, 2012]. The kinetic equations is given in the body frame, which is fixed to the frame of the UAV. The kinematics equations is given as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \mathbf{R}(\mathbf{\Theta})_{Body}^{NED} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.1a}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{T}(\mathbf{\Theta}_{nb}) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.1b}$$

where $\mathbf{R}(\mathbf{\Theta})_{Body}^{NED}$ is the rotation matrix from the body frame to the NED frame, with $\mathbf{\Theta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$. The transformation matrix $\mathbf{T}(\mathbf{\Theta}_{nb})$ is given in [Fossen, 2011] as:

$$\mathbf{T}(\mathbf{\Theta}_{nb}) = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \tag{2.2}$$

The kinetic equations is given as:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \mathbf{R}(\mathbf{\Theta})_{NED}^{Body} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \frac{1}{2}\rho V_a^2 S \mathbf{R}(\alpha)_{Stability}^{Body} \begin{bmatrix} F_{Drag} \\ 0 \\ F_{Lift} \end{bmatrix} \tag{2.3a}$$
$$+ \frac{1}{2}\rho V_a^2 S \begin{bmatrix} 0 \\ C_y(\beta, p, r, \delta_a, \delta_r) \\ 0 \end{bmatrix} + \frac{1}{2}\rho S_{Prop} C_{Prop} \begin{bmatrix} (K_{Motor}\delta_t)^2 - V_a^2 \\ 0 \\ 0 \end{bmatrix}$$
$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} C_L(\beta, p, r, \delta_a, \delta_r) \\ C_M(\alpha, q, \delta_e) \\ C_N(\beta, p, r, \delta_a, \delta_r) \end{bmatrix} + \begin{bmatrix} -k_{T_p}(K_\Omega \delta_t)^2 \\ 0 \\ 0 \end{bmatrix} \tag{2.3b}$$

where ρ is the air density in kg/m^3, S is the platform area of the MAV wing, C_i is nondimensional aerodynamic coefficients and V_a is the speed of the MAV through the surrounding air. α and β is the attack and side slip angle respectfully. F_{Drag} is the drag force acting on the fuselage, and F_{Lift} is the lift force. $\mathbf{R}(\alpha)_{Stability}^{Body}$ is the rotation matrix from the stability frame to the body frame. The stability frame is orientated with respect to the MAV movement through the surrounding air, which is defined as a standard rotation around the y-axis. S_{Prop} is the area swept out by the propeller, and K_{Motor},K_{T_p} and K_Ω is propeller specific constants. The control surface on the MAV is defined into two groups; the wings and the rudder. On the rudder δ_e controls the elevator deflection and δ_r the rudder deflection. For the wings δ_a is the control input from the aileron deflection. The control input for the control input is δ_t.

2.2 Landing path modelling

A landing path is a path following problem where the minimum requirement is that the path is connected, and is flyable. The connection level can be described by the paths smoothness. Parametric continuity is denoted C^n were n is the degree of smoothness. The order of n implies that the n first parametric derivatives match at a common point for two subsequent paths [Barsky and DeRose, 1989]. Geometric continuity is a relaxed from of parametric continuity in witch discontinuousness in speed is allowed. A table 2.1 of geometric and parametric continuity lists the requirement for each smoothness level, which is based definitions presented in [Barsky and DeRose, 1989]. Geometric continuity is sufficient for a path following system, which is the main focus of this thesis. Geometric continuity is denoted as G^n were n is the order of continuity.

Geometrical smoothness level	**Description**
G^0	All subpaths are connected
G^1	The path-tangential angle is continuous
G^2	The center of curvature is continuous
Parametric smoothness level	**Description**
C^0	All subpaths are connected
C^1	The velocity is continuous
C^2	The acceleration is continuous

Table 2.1: Smoothness definitions

The definition used for path in this thesis is equation 1.2 in [Tsourdos et al., 2010] which state:

$$P_s(x_s, y_s, z_s, \theta_s, \psi_s) \xrightarrow{r(\varpi)} P_f(x_f, y_f, z_f, \theta_f, \psi_f) \tag{2.4}$$

where the subscripts s and f denotes the start pose and finish pose respectfully with $r(\varpi)$ as the path.

2.2.1 Straight lines

The simplest form on path is a straight line between P_s and P_f. For simplicity the path is reduced to a 2 dimensional case, where the straight line is given as

$$x(\varpi) = a_x\varpi + b_x \tag{2.5a}$$
$$y(\varpi) = a_y\varpi + b_y \tag{2.5b}$$

with $\varpi \in [0, 1]$, where ϖ has not necessary a physical meaning. Then the parametrisation of the straight line is:

$$P(0) = \begin{bmatrix} x(0) \\ y(0) \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{2.6a}$$
$$P(1) = \begin{bmatrix} x(1) \\ y(1) \end{bmatrix} = \begin{bmatrix} a_x + b_x \\ a_y + b_y \end{bmatrix} = \begin{bmatrix} x_f \\ y_f \end{bmatrix} \rightarrow \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} x_f - b_x \\ y_f - b_y \end{bmatrix} \tag{2.6b}$$

The tangential vector for a straight line is given as:

$$\psi(\varpi) = \operatorname{atan} 2(a_x, a_y) \tag{2.7}$$

with it's derivative:

$$\dot{\psi}(\varpi) = 0 \tag{2.8}$$

A path constructed by straight lines is G^0, however since the tangential vectors derivative is zeros, it is discontinuous between two line segments with different heading and therefore not G^1[TODO: LEGG INN FIGUR SOM VISER DISKONINUITETEN]. The disadvantage with a path which is G^0 is that large discontinuity between two tangential vectors will cause problem for a control system.

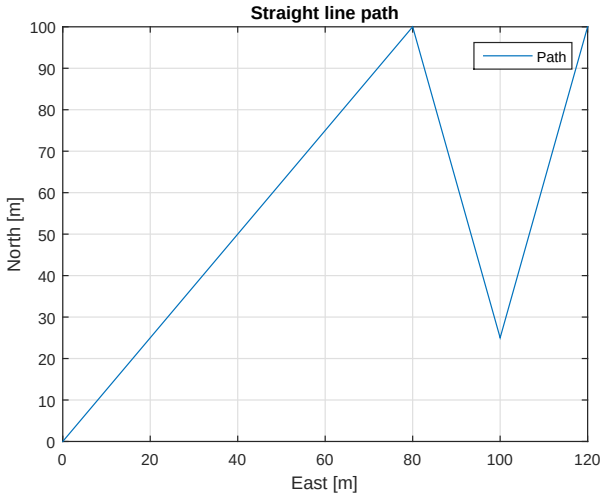


Figure 2.1: Straight line path

2.2.2 Dubins path

An alternative to a straight line path is a path constructed by straight lines and circle. Such a path is Dubins path[Dubins, 1957], which showed that the shortest possible path for a particle that moved with unit speed with maximum curvature would consist of two circles and a straight line which is tangential to both circles. A disadvantage with Dubins path is that the curvature is discontinues, which gives a path from P_s to P_f with smoothness level of G^1.

A Dubins path that is constructed where the final orientation is fixed has four different ways to be constructed, which is determined by the rotation directions. The four types of Dubins path that is used in this thesis is given in table 2.2.

Right to Right
Right to left
Left to Right
Left to left

Table 2.2: Turning direction for Dubins path with fixed final orientation

The equations that is used to construct the path is found in [Tsourdos et al., 2010] section 2.2.1, with a constructed path shown in figure 2.2.

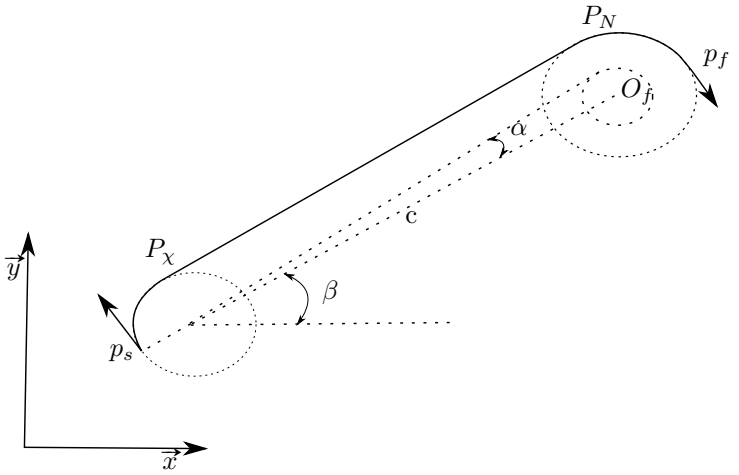


Figure 2.2: Dubins path

The first step if to determine the start and final turning circle center. The center is found with the equations:

$$X_{cs} = X_s - R_s \cos(\psi_s \pm \frac{\pi}{2}) \tag{2.9a}$$

$$Y_{cs} = Y_s - R_s \sin(\psi_s \pm \frac{\pi}{2}) \tag{2.9b}$$

$$X_{cf} = X_f - R_f \cos(\psi_f \pm \frac{\pi}{2}) \tag{2.9c}$$

$$Y_{cf} = Y_f - R_f \sin(\psi_f \pm \frac{\pi}{2}) \tag{2.9d}$$

where R_s and R_f is the radius of the start and final turning circle respectfully, with ψ_s and ψ_f the start and final heading. The centres for the start and final turning circle is defined as:

$$\mathbf{O}_{cs} = \begin{bmatrix} X_{cs} \\ Y_{cs} \end{bmatrix} \tag{2.10}$$

$$\mathbf{O}_{cf} = \begin{bmatrix} X_{cf} \\ Y_{cf} \end{bmatrix} \tag{2.11}$$

Continuing the centres O_{cs} and O_{cf} is connected with a centreline c, where the length is given as:

$$|c| = ||\mathbf{O}_{cs} - \mathbf{O}_{cf}||_2 \tag{2.12}$$

where $||\cdot||_2$ is the second norm. Continuing the arc exit and entry point for the start and final circle is calculated by first applying the equations:

$$\alpha = \arcsin\left(\frac{R_f - R_s}{|c|}\right) \tag{2.13a}$$

$$\beta = \arctan\left(\frac{Y_{cf} - Y_{cs}}{X_{cf} - X_{cs}}\right) \tag{2.13b}$$

where α is the angle between the length of the center line between the two circles, and the length of the line from the start circle to the exit tangent point. β is the angle of the center line with respect to the inertial frame. The exit and entry tangent point is found with the use of table **??**.

	Turn angle
ϕ_{right}	$\alpha + \beta + \frac{\pi}{2}$
ϕ_{left}	$\beta - \alpha + \frac{3\pi}{2}$

With the angle of the exit and entry tangent point the point is given as:

$$x_{P_\chi} = x_{cs} + R_s\cos(\phi) \tag{2.14a}$$

$$y_{P_\chi} = x_{cs} + R_s\sin(\phi) \tag{2.14b}$$

$$x_{P_N} = x_{cf} + R_f\cos(\phi) \tag{2.14c}$$

$$y_{P_N} = x_{cf} + R_f\sin(\phi) \tag{2.14d}$$

which is used to define the exit and entry points as:

$$\mathbf{P}_\chi = \begin{bmatrix} x_{P_\chi} \\ y_{P_\chi} \end{bmatrix} \tag{2.15a}$$

$$\mathbf{P}_N = \begin{bmatrix} x_{P_N} \\ y_{P_N} \end{bmatrix} \tag{2.15b}$$

The length of the path is calculated in three parts. The first the is the arc length from the start pose to the exit tangent point, then the length of the straight line before the arc length from the entry point to the final pose. The length of the path is given as:

$$d = R_s\phi_s + d_t + R_f\phi_f \tag{2.16}$$

where $d_t = ||\mathbf{P}_N - \mathbf{P}_\chi||_2$, ϕ_s and ϕ_f is the arc angle for the start and final circle respectfully.

2.3 Position estimation RTK-GPS

In [Misra and Enge, 2011] section 7.2.2 Real Time Kinematic GPS (RTK-GPS) is defined as a rover that receive raw measurements from a reference receiver which is transmitted over a radio link, with a key feature that the rover is able to estimate the integer ambiguities while moving. The reference receiver is usually defined as a base station, and the integer ambiguity is the uncertainty of the number of whole phase cycles between the receiver and a satellite. With the measurements from the base station the rover is able to calculated the distance between itself and the base station, where the distance is referred to as a baseline. The length of the baseline affect the accuracy of the RTK-GPS solution, due to increased effect of atmospheric disturbance, which is further explain in 2.3.1. However with a short baseline, e.g. $1 - 2km$, the atmospheric condition can be considered equal for the base station and the rover, which keeps the solution at centimetre level accuracy. The concept of RTK-GPS is depicted in figure 2.3.

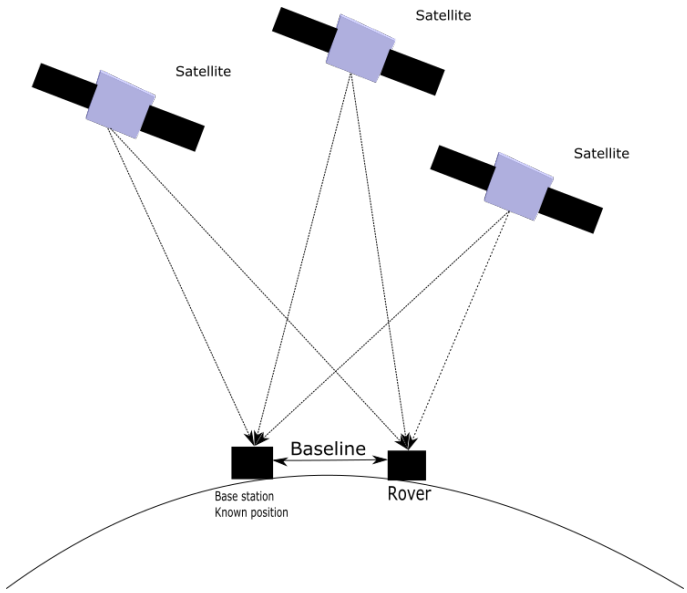


Figure 2.3: Concept figure of Real Time Kinematic GPS (RTK-GPS)

The ability for the rover to resolve the integer ambiguity is a key feature in RTK-GPS. A well used method was purposed in the article [Teunissen, 1994] which decorrelate the integer ambiguities such that a efficient computation of the least square estimate can be performed. The search method is further explained in [Teunissen, 1995]. A estimate of the integer ambiguity with sufficient high degree of certainty is referred to as a FIX solution, otherwise the solution is degraded to

FLOAT where the integer ambiguity is allowed to be a decimal or a floating point number. When the solution is categorised as FIX the accuracy of the solution is considered on centimetre level, while with a FLOAT solution the accuracy is at a decimetre level. However when a FIX solution is lost, the solution accuracy will not imminently degrade to decimetre level.

In RTK-GPS the position of the base station must be resolved. This can be achieved by either knowing the position beforehand, which is defined as a kinematic configuration. If the base station position is unknown the RTK-GPS solver calculates the position on the fly, which is defined as a moving baseline configuration. The unknown is then calculated as a standalone Global Navigation Satellite System (GNSS) receiver, with the accuracy that entails. Therefore the RTK-GPS system with a moving baseline configuration can never have better global accuracy then what it will get with a single receiver. The advantage with the moving baseline configuration is that RTK-GPS can be used to find the relative position between two dynamical system using GNSS in real time. This will be the case in automatic ship landing system, where the base station is on a ship, thus must be allowed to move. The advantage with kinematic mode is that it can give a more accurate position estimate, where the relative position of the rover can be given in either the North East Down (NED) or East North Up (ENU) frame.

2.3.1 Error sources

In order to get high accuracy in the position estimation the different error sources must be identified and removed if possible. This section will identify some of the most significant error sources that can affect the GNSS signal, and how to remove or mitigate them in the estimation.

Clock error

There is drift in both the satellite clock and the receiver clock. The atomic clock in the satellites makes the clock drift negligible from the user perspective. The receiver clock tend to drift, and if not taken into account will cause large deviations in the position estimate from the true position. This error is remove by including a fourth satellite in the position computation. The satellite clock error is given in the satellite message.

Ionospheric and tropospheric delays

When the GPS signals travel though the atmosphere there will be a delay caused by the different atmospheric layers. The atmosphere change the velocity of wave propagation for the radio signal, which results in altered transit time of the signal.

Ionospheric delay Gas molecules in the ionosphere becomes ionized by the ultra-violet rays that is emitted by the sun, which release free electrons. These electron can influence electromagnetic wave propagation, such as GNSS signals. In [Vik, 2014] section 3.5.1 it's stated that the delay caused by the ionosphere usually is in the order of $1 - 10$meters. The error can be mitigated by using a double frequency receiver, or by applying a mathematical model to estimate the delay. Both those methods are with a single receiver, however by including a second receiver in a network, e.g. RTK-GPS, the GNSS solution system can assume that both receiver receive signal in the same epoch, which means that the signals have experienced the same delay. The rover is then able to remove the error induced from ionospheric disturbance.

Tropospheric delay The tropospheric delay is a function of the local temperature, pressure and relative humidity. The effect of tropospheric delay can vary from 2.4 meters to 25 meters depending on the elevation angle of the satellites,[Vik, 2014] section 3.5.1. The error can be mitigated by applying a mathematical model to estimate the tropospheric delay, or by using a elevation mask can remove all satellites with a elevation angle bellow a certain threshold. Similar to ionospheric delay, tropospheric delay can be removed when using two receivers in a network by assuming that the single received by both receivers has experienced the same delay. The tropospheric delay is a function of the local temperature, pressure and relative humidity. The effect of tropospheric delay can vary from 2.4 meters to 25 meters depending on the elevation angle of the satellites,[Vik, 2014] section 3.5.1. The error can be mitigated by applying a mathematical model to estimate the tropospheric delay, or by using a elevation mask can remove all satellites with a elevation angle bellow a certain threshold. Similar to ionospheric delay, tropospheric delay can be removed when using two receivers in a network by assuming that the single received by both receivers has experienced the same delay.

Multipath

One of the primary source of error in in a GNSS receiver is multipath. Multipath happens when the satellite signal is reflected by a nearby surface before if reach the GNSS antenna. The delay introduced in the signal can make the receiver believe that its position is several meters away form its true position. The easiest way to mitigated multipath is to place the antenna at a location with open skies, with no tall structures nearby. The effect can also be mitigated by choosing a antenna with good multipath rejection capability.

Multipath error uncorrelated between receivers, thus the local receiver must be able to correct for multipath error locally.

Chapter 3

Applied software and hardware

3.1 LSTS toolchain

The software that the system is based on was developed by the Underwater Systems and Technology Laboratory (LSTS), which is called the LSTS toolchain [Pinto et al., 2013]. The toolchain was developed for support of networked heterogeneous air and ocean vehicle systems over wireless network. The toolchain contain four different modules, namely IMC, DUNE, NEPTUS and Glued.

3.1.1 IMC

IMC [Martins et al., 2009] is design to enable interconnections between systems of vehicles, sensors and human operators, which enable the pursuit of common goal by cooperatively exchange real-time information about the environment and updated objectives. The message protocol is oriented around the message, which abstracts hardware and communication heterogeneity with a provided shared set of messages that can be serialized and transferred over different means. The IMC protocol is defined in a single eXtensible Markup Language (XML) document, which simplify the definition of exiting messages and the creation of new messages. A single XML document ease communication between two node when both node use the same document for message definition.

3.1.2 Dune

DUNE (DUNE Uniform Navigation Environment) is a runtime environment for unmanned systems on-board software written in C++. DUNE is capable to interact with sensors, payload and actuators, in addition to communication, navigation, control, manoeuvring, plan execution and vehicle supervision. The software separate operations into different task that each has there own thread of execution. DUNE

apply a message bus that is responsible for forwarding IMC message from the producer to all registered receivers, which is the only way different DUNE tasks is communicating.

A DUNE task is enabled through a configuration file, where the user can choose in which profile the task should be enabled in. The different profile configuration in DUNE allows for testing the same system used in a hardware setting with a simulator.

3.1.3 Neptus

Neptus is a Command and Control software which is used to command and monitor unmanned systems that is written in Java. Neptus is able to provide coherent visual interface to command despite the heterogeneity in the controlled system that it is interacting with. This allow the operator to command and control unmanned system without the need to dwell into specific command and control software in the unmanned system. The main communication channel for Neptus is IMC, which makes it interoperable with DUNE or other IMC- based peer.

Neptus is able to do MRA (Mission Review and Analysis) after a mission is finished. In the MRA phase Neptus analyse the IMC logs that is collected by e.g. DUNE, such that the result from a completed mission can be presented. In addition Neptus mission review is able to create output files of the log that can be analysed in third party software like Matlab.

3.1.4 Glued

Glued is a minimal Linux operating system distribution, and design with embedded system in mind. It is platform independent, easy to configure and contain only the necessary packages to run on a embedded system. This makes GLUED a light and fast distribution, which is ideal for a on-board operating system for a unmanned system where payload size is normally limited. GLUED is configured through a single configuration file that which can be created for a specific system. A advantage with Glued is that it can be cross-compiled, which allows for compilation of software before it's transferred to the embedded computer.

3.2 RTKLIB

Real-Time Kinematic Library (RTKLIB)[Takasu and Yasuda, 2009] is a open source program package for standard and precise positioning with GNSS developed by T. Takasu. Real-Time Kinematic Library (RTKLIB) can be configured to apply RTK-GPS, such that raw GNSS data is used estimate the relative position of the rover with respect to the base station in real time. Figure 5.1 shows how RTKLIB

can be used in a RTK-GPS mode. The two main modules here is str2str and rtkrcv. The version of RTKLIB used in this thesis is RTKLIB2.4.2 [RTKLIB].

Rtklib is configured as a moving baseline, where the baseline between the base station and the rover is accurately estimated with centimeter level accuracy. However the concept of moving baseline indicates that the base station is allowed to move, which require continues calculation of the base station position as a standalone GPS. Therefore the error sources that are mitigated in the RTK-GPS solution is present in the GPS position of the base station. The moving baseline configuration is used since a fixed base station location is not known, and a navigation system that should be used at sea will not have a fixed base station location present.

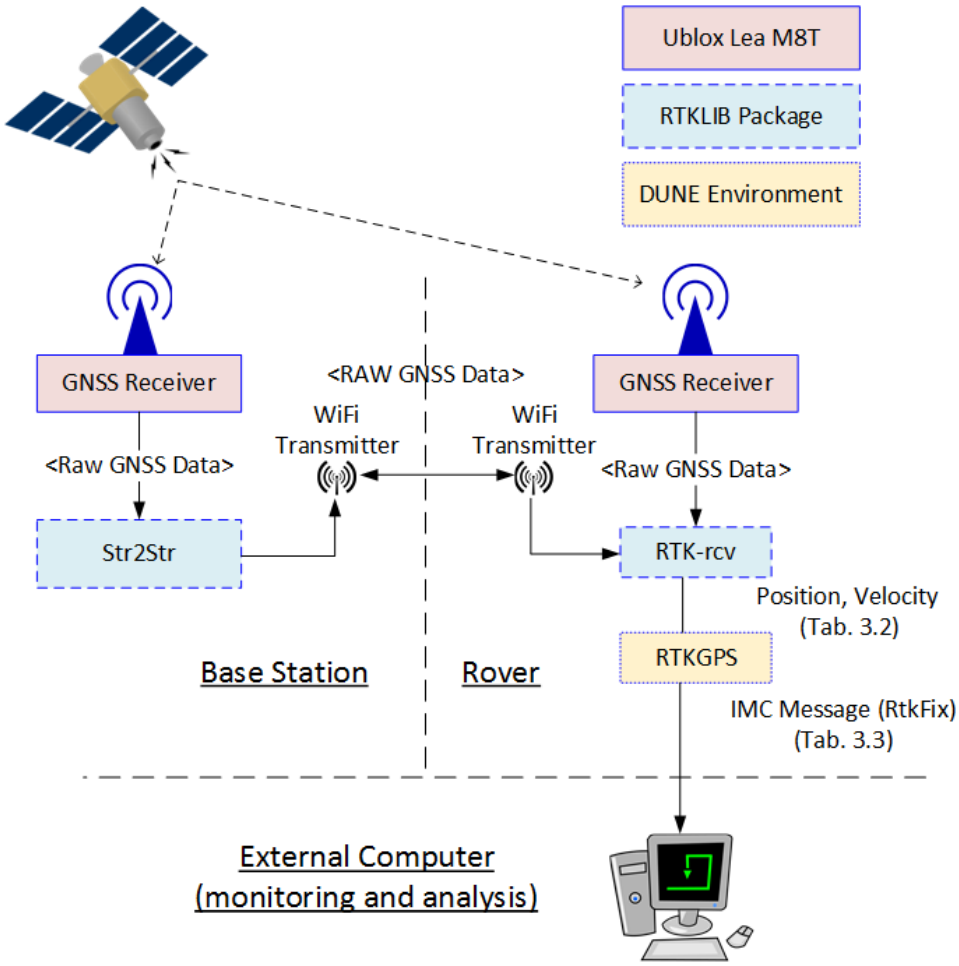


Figure 3.1: The communication structure of RTKLIB

3.3 Pixhawk

3DR Pixhawk is a high-performance autopilot suitable for fixed wing multi rotors, helicopter and other robotic platform that can move. The Pixhawk system comes complete with GPS, imu, airspeed sensor and magnetometer.

3.4 Ardupilot

Ardupilot is an open-source unmanned aerial vehicle platform, able to control fixed wing UAV and multicopters. Ardupilot is used for low level control of the UAV, and is the software that runs on the Pixhawk. Ardupilot is able to communicate to third party software e.g Dune. Ardupilot uses the sensors in the Pixhawk to calculate the position, velocity and attitude of the UAV, which is sent to DUNE.

3.5 JSBsim

JSBSim [Berndt, 2004] is an open-source flight dynamic model that is able to simulate a physical model of an arbitrary aircraft without the need of specific compiled and linked program code. The simulator is design such that a third party software e.g. Ardupilot can expose the model to external forces and moments. This enable Software In the Loop (SIL) testing of system that is able to run in a hardware configuration with only minor configuration alteration.

3.6 X8 and nest payload

The Skywalker X8 is fixed wing UAV in a flying wing configuration, which indicate that the UAV has no tail and clear distinction between the wings and fuselage. The X8 is a popular choice for experimental missions at the UAV-lab at the Deparment of Engineering Cybernetic since it's durable, cheap and enough space to carry experimental payload. The X8 is used to test the landing path discussed in this thesis, however the navigation system has been tested in both the X8 and a multicopter system.

The hardware configuration used in the X8 and nest systems is based on the proposed hardware in the paper [Zolich et al., 2015]. The X8 and the nest systems are installed with a BeagleBone embedded computer with the Glued operating system, which is used to run the Dune system, as well as rtklib. The autopilot used in the X8 is a 3DR Pixhawk with ArduPilot ArduPlane software. For the RTK-GPS system Ublox Lea M8T GNSS receivers [U-blox, a,b] are connected to the BeagleBone with uart cable, which is configured with a output rate of 10Hz. The antenna used in the X8 is a Maxtena M1227HCT-A-SMA L1/L2 GPS-GLONASS Active Antenna

[Maxtena], and the antenna used in the base station is a Novatel GPS-701-GG [Novatel].

The communication between the X8 and the nest systems is done with Ubiquiti M5 rocket [roc] radios, where the communication between each unit can be done with TCP/UDP/IP.

Chapter 4

Path and Navigation

4.1 Path system

The path system is designed to enable UAV landing in both a stationary and a moving net. The path is created in two main stages. The first is the creation of the landing path, which is defined as a straight line along the heading of the net, as shown in figure 4.1. The second stage is the approach path, which apply a lateral Dubins path and longitudinal straight line path to create a path that ensures that the UAV is able to enter the landing path at the correct height and attitude.

4.1.1 Landing Path

The landing path is inspired by the work done in [Skulstad and Syversen, 2014] where waypoint was used to create a straight line path towards the net. This method proved successful, and since the UAV descent towards the net should be as controlled as possible only small angles is used when transitioning between way-points. The straight line path is constructed relative to the net as shown in figure 4.1, with

way-points given as:

$$\mathbf{WP4} = \begin{bmatrix} -a0 \\ 0 \\ h_{nc} + a1\tan(\gamma_a) \end{bmatrix} \tag{4.1a}$$

$$\mathbf{WP3} = \begin{bmatrix} a1 \\ 0 \\ h_{nc} - a1\tan(\gamma_a) \end{bmatrix} \tag{4.1b}$$

$$\mathbf{WP2} = \mathbf{WP3} + \begin{bmatrix} a2 \\ 0 \\ a2\tan(\gamma_d) \end{bmatrix} \tag{4.1c}$$

$$\mathbf{WP1} = \mathbf{WP2} + \begin{bmatrix} a3 \\ 0 \\ 0 \end{bmatrix} \tag{4.1d}$$

were the description of the parameters used is given in table 4.1. The net is placed between the fourth and third way points such that transitional behaviour do not occur during the finale stage of the net landing. In addition the path has been made with the assumption that the γ_a and γ_d is considered small. This assumption is made to ease the demand of the controllers used in the landing system.

Parameter	**Description**
h_{nc}	The height from ground to the net center
$a0$	The distance behind the net
$a1$	The distance in front of the net
$a2$	The length of the glide slope
$a3$	The length of the approach towards the glide slope
γ_a	The net attack angle
γ_d	The glide slope angle

Table 4.1: Net approach parameters

The way point vectors are rotated into the NED frame by a rotation around the z-axes.

$$\mathbf{WP}^n = \mathbf{R}(\psi_{net})\mathbf{WP}^b \tag{4.2}$$

were ψ_{net} is the heading of the net, and $\mathbf{R}(\psi_{net})$ is the rotation matrix around the z-axis.

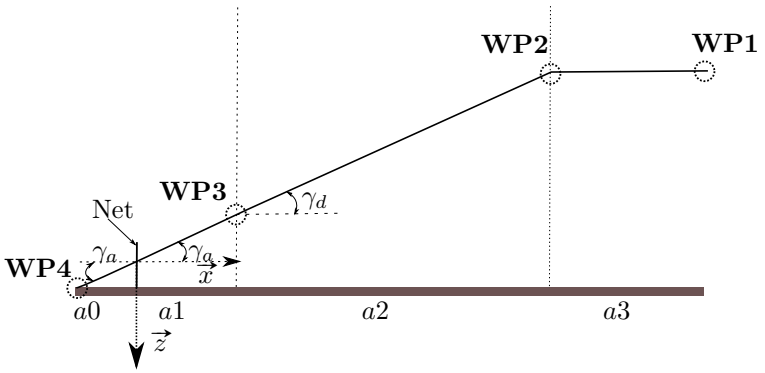


Figure 4.1: The virtual runway

4.1.2 Approach path

The landing path is separated into two parts, which is a lateral and longitudinal path. The path is designed to ensure that the UAV follows the path along the virtual runway it must be at the correct height with the correct attitude from any initial position. In addition it's desirable that the decent angle is kept small to ease the strain on the control system, and reduce chance of overshoot of way-points.

Lateral path

The lateral path is designed as a Dubins path, where the initial position is the start pose, P_s, and the final pose P_f is the start of the landing path. Dubins path was chosen due to it's simplicity in description and it fulfils the requirement that it's smooth enough for path following.

The lateral path is constructed following the equations in section 2.2.2, however the rotation direction in each circle must first be determined. The desired behaviour is to find the shortest path of the four different variants given in table 2.2. The shortest path is determined by calculating the length of each variants, where the shortest is chosen. The resulting the path is shown in figure 4.2.

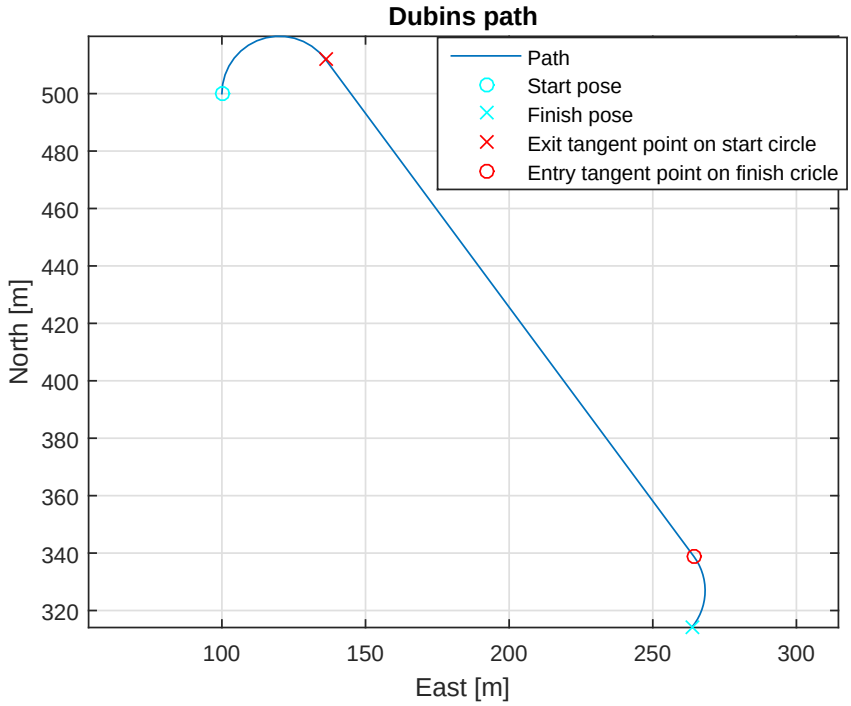


Figure 4.2: Lateral Dubins path

$$\psi_0 = \begin{cases} \operatorname{atan2}(Y_s - Y_{cs}, X_s - X_{cs}) & \text{if start circle} \\ \operatorname{atan2}(Y_{P_N} - Y_{cf}, X_{P_N} - X_{cf}) & \text{otherwise} \end{cases} \tag{4.3a}$$

$$\psi_1 = \begin{cases} \operatorname{atan2}(Y_{P_\chi} - Y_{cs}, X_{P_\chi} - X_{cs}) & \text{if start circle} \\ \operatorname{atan2}(Y_f - Y_{cf}, X_f - X_{cf}) & \text{otherwise} \end{cases} \tag{4.3b}$$

$$\psi_{max} = \begin{cases} -|\psi_1 - \psi_0| & \text{if counter clockwise rotation and } \psi_1 - \psi_0 \leq 0 \\ -(2\pi - |\psi_1 - \psi_0|) & \text{if counter clockwise rotation and } \psi_1 - \psi_0 > 0 \\ |\psi_1 - \psi_0| & \text{if clockwise rotation and } \psi_1 - \psi_0 \geq 0 \\ (2\pi - |\psi_1 - \psi_0|) & \text{if clockwise rotation and } \psi_1 - \psi_0 < 0 \end{cases} \tag{4.4}$$

where $psi_1 - \psi_0 \in (-\pi, \pi]$.

$$h = \frac{d_{arc}}{R} \tag{4.5a}$$

$$N = \left\lceil \frac{\text{sign}(\psi_{max})\psi_{max}}{h} \right\rceil + 1 \tag{4.5b}$$

$$h = \text{sign}(\psi_{max})h \tag{4.5c}$$

where h is arc angle step and N the total number of steps in the arc.

$$\psi(\varpi) = \begin{cases} \psi_{max} & \varpi = N - 1 \\ \varpi h & \text{otherwise} \end{cases} \tag{4.6}$$

where $\varpi = 1, ..., N - 1$.

$$\mathbf{r}(\varpi) = \begin{bmatrix} \mathbf{O}_c \\ 0 \end{bmatrix} + R \begin{bmatrix} \cos(\psi_0 + \psi(\varpi)) \\ \sin(\psi_0 + \psi(\varpi)) \\ 0 \end{bmatrix} \tag{4.7}$$

Longitudinal path

The longitudinal path is designed as a straight line along the lateral path, which results in a spiral path in the arcs created by the lateral path. The path hold a constant decent angle until the correct height is reached, which is defined as the start height for the virtual runway. The resulting height profile of the approach path becomes a straight line path shown in figure 4.3. The longitudinal path is given as:

$$r(i+1) = r(i) + \begin{bmatrix} 0 \\ 0 \\ ||p(i)||_2 \tan(\gamma_d) \end{bmatrix} \tag{4.8}$$

where $r(i)$ is the landing path and $p = \begin{bmatrix} x(i) & y(i) \end{bmatrix}^T$ is the lateral path. During each iteration the algorithm checks if the next step can reach the correct height with a decent angel greater or equal to the maximum decent angel γ_d. In the case where the the height at the end of the lateral path is not the same height as the start height for the virtual runway, the landing path will enter a spiral which decent towards the correct height. The spiral is centred in the same circle that is used to create the arc for the final turn in the lateral path. The spiral is exited when the correct height is reached, and a arc is created from the spiral exit point towards the lateral path final point.

The height profile of the landing path is shown in figure 4.3,with the resulting path connected to the virtual runway shown in figure 4.1.

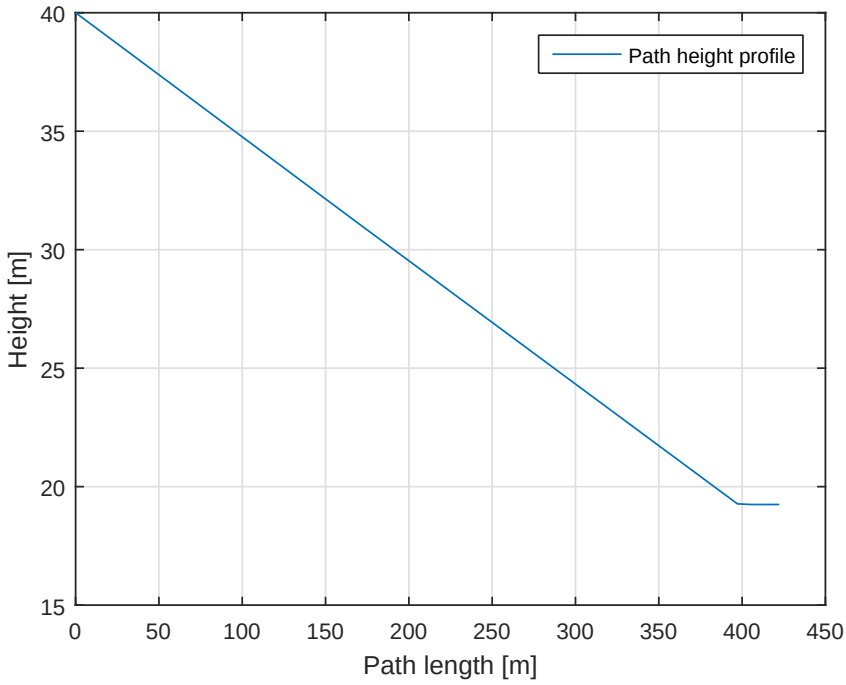


Figure 4.3: Height profile of the landing path

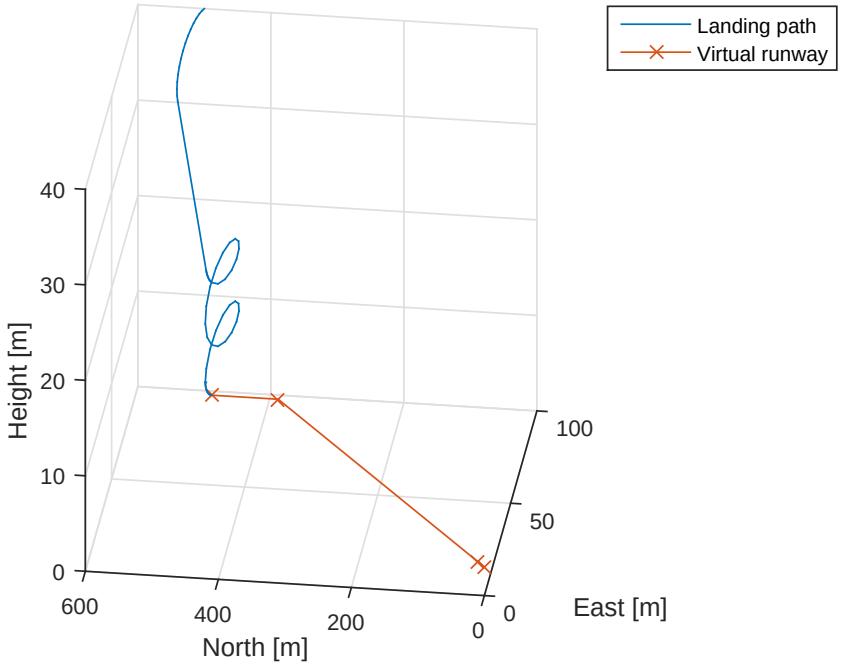


Figure 4.4: Landing path connected to the virtual runway

4.2 Navigation system

The navigation system apply RTK-GPS for position and velocity measurement, which provide higher position accuracy then a standalone GPS. However the RTK-GPS is subject to drop out, which create a situation where the navigation system should switch to standalone GPS or wait for the RTK-GPS to return. A state machine has been created to handle the state switching between RTK-GPS and the external navigation data, which is navigation data from the Pixhawk. This is handled in two steps. The first is a short loss compensator stage, where the backup standalone GPS is compensated to get a position solution closer to the RTK-GPS solution. The second stage fully disconnect the RTK-GPS.

4.2.1 Navigation state

The navigation system consists of five states, which is controlled by a state machine. The state transitions are determined by what's available and what the operator has specified should be used. The output from the state machine is the IMC message EstimatedState, which is the only state source used in the Dune system. The state

machine also dispatch a IMC message that informs which the source used in the navigation system, and which sources is available. Currently only the RTK-GPS system is considered as a internal system, however this can be expanded to include other sensors used in the Dune system.

The input to the navigation system is IMC message ExternalNav and GpsRtkFix. The ExternalNav message is the primary state source when the RTK-GPS is not in use, and it's receive the state information from the Pixhawk mounted in the X8.

During a short loss of the RTK-GPS the position solution in the ExternalNav message is compensated to avoid sudden change in position. The short loss compensator is explain further in 4.2.1. The short loss system is implemented to avoid drop out of the RTK-GPS when a message is delayed, or its struggling due to the dynamic behaviour of the UAV.

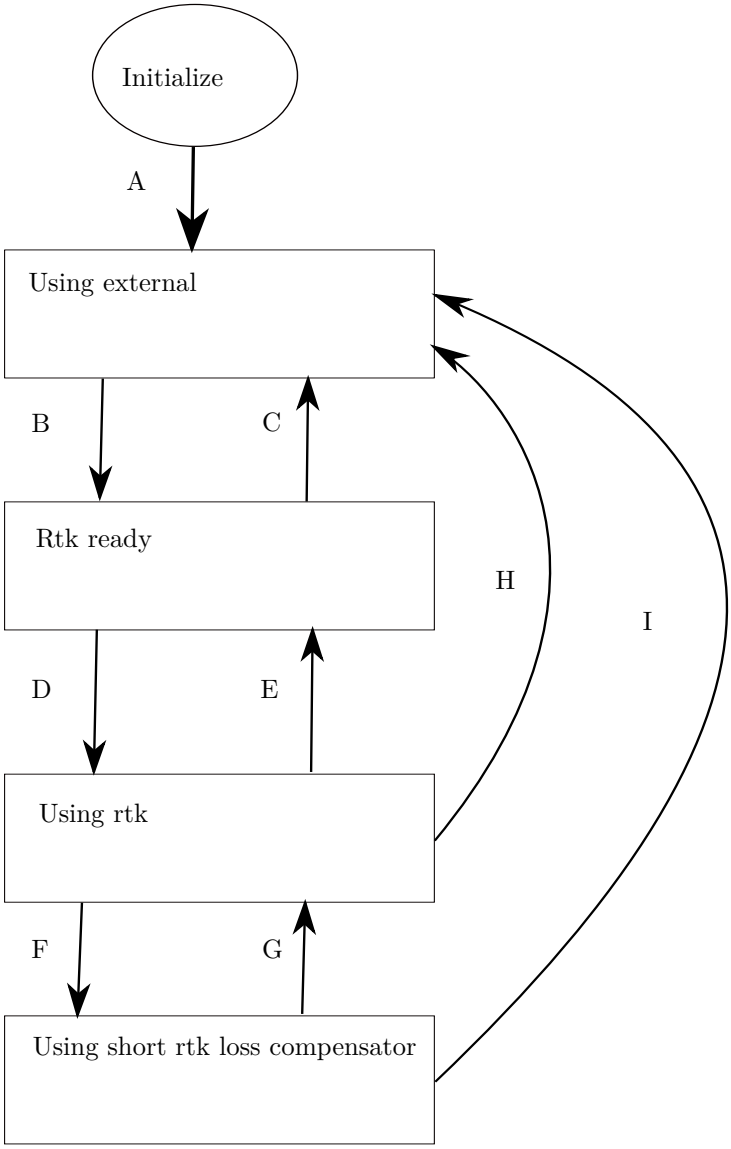
The state machine is depicted in figure **??**, with the edge description given in table 4.2.

Edge	**Event**	**Guard**
A	Event: Received External Nav message	None
B	Time out: Fix RTK-GPS solution for x seconds	None
C	Time out: x seconds since last valid GpsFixRtk	None
D	Flag: Using Rtk is set true	None
E	Flag: Using Rtk is set false	None
F	Time out: x seconds since last valid GpsFixRtk Event: Received GpsRtkFix with $type == None$	Short loss compensator: Enabled
G	Event: Received valid GpsFixRtk message	None
H	Time out: x seconds since last valid GpsFixRtk Event: Received GpsRtkFix with $type == None$	Short loss compensator: Disabled
I	Time out: x seconds since last valid GpsFixRtk	None

Table 4.2: Net approach parameters

Short loss of RTKGPS

In the event of RTK-GPS drop out a offset can be added to the position solution in order to prevent a sudden change in position. The offset is defined as the average difference between the N latest position solution from the RTK-GPS and the external


Initialize
A
Using external
B
C
Rtk ready
H
I
D
E
Using rtk
F
G
Using short rtk loss compensator

State	**Description**
Initialize	The task starting up
Using external	The navigation task apply the external navigation source in the state message
RTK ready	The RTK-GPS is ready for use, however the external navigation source is still used
Using RTK	The navigation task apply the RTK-GPS in the state message
Using short RTK loss compensator	The navigation task apply the external navigation source with a compensation term to reduce the effect of RTK-GPS loss.

navigation system:

$$offset = \frac{1}{N}\sum_{n=0}^{N}(RTKGPS(n) - External(n)) \tag{4.9}$$

where the RTK-GPS solution is displaced into the External nav system. However during the implantation it was discovered that the standard displace function in the DUNE literary was inaccurate in correctly calculating the hight of a offset point. The problem was that the geodetic latitude calculation was calculate assuming that the Earth has a spheric shape. This was solved by creating a new displace function where the geodetic latitude is calculated in according to

Chapter 5

Implementation

This chapter contain the technical specification of the navigation system and landing path system, in addition to the payload installed in the X8 and both nests.

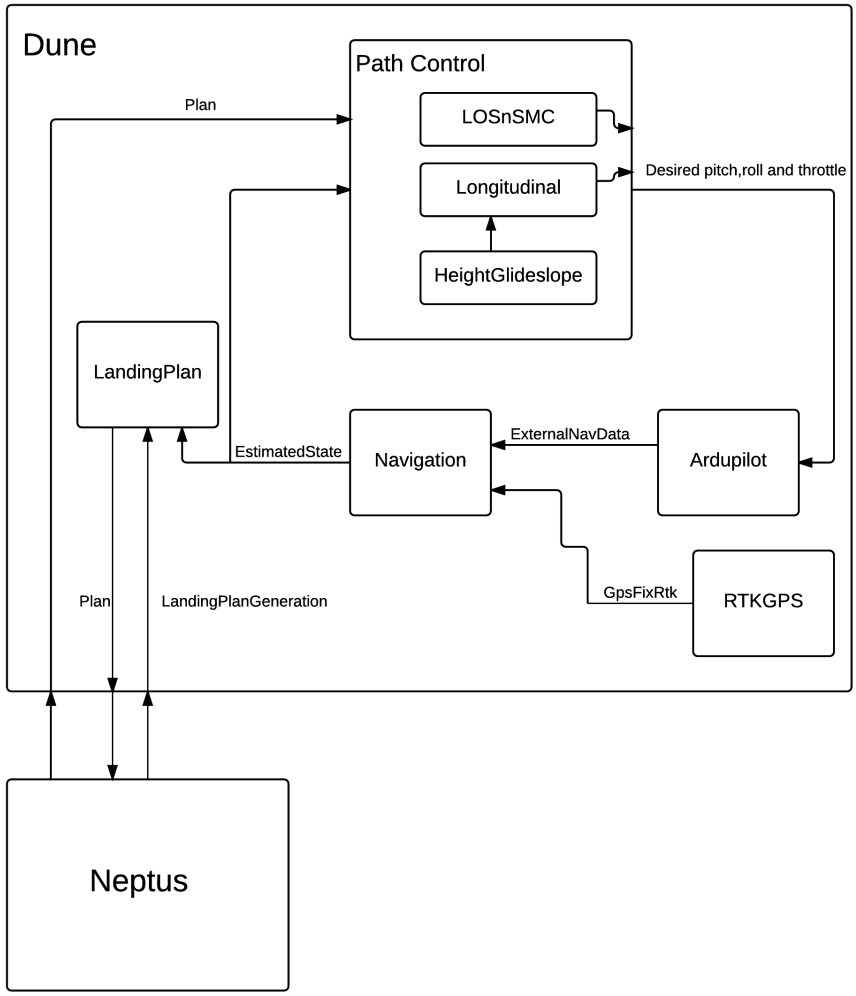


Figure 5.1: A simplified figure of the Dune auto land system

5.1 Landing path

The landing path is specified through the Neptus plug-in LandmapLayer, which allow for real time setting of the parameters used to create the landing path. In addition the plug-in display the position of the net in the graphical map interface which is a part of Neptus. The plug-in dispatch the IMC message, LandingPlanGeneration, which has been created specifically for landing path purpose.

The IMC LandingPlanGeneration message is dispatch to the Dune system, where

Parameter name	**Action**
Automatic (boolean)	If true a standard path where the shortest Dubins path is chosen. Otherwise a user specific path is chosen
Start circle turning counter clockwise (boolean)	If true the start arc is created such that the turning direction is counter clockwise. Otherwise clockwise. Require Automatic==false
Finish circle turning counter clockwise (boolean)	If true the finish arc is created such that the turning direction is counter clockwise. Otherwise clockwise. Require Automatic==false
Wait at loiter (boolean)	If true a unlimited loiter is included in the path before the path continue with the path along the virtual runway.

it's picked up by the Dune task LandingPlan. This trigger the generation of the landing path, and depending on the configuration of the Dune task and desired setting in the LandingPlanGeneration message different landing path is generated. The different variants is categorised into two groups; configuration of the Dubins path, and configuration of the virtual runway. The configuration option that result in different variants of the landing path is given in table **??**. In addition the Dune task can be configured for a dynamical landing. When configured as dynamical it's assumed that an other task is responsible for the final landing part. Therefore the vertical runway is not part of the plan, however the first part of the path is still included.

A design choice allows for user determined rotation direction. By setting the "automatic" flag to false in the IMC message LandingPlanGeneration the rotation direction is determined by the flags "startCounterClockwise" and "finishCounterClockwise".

When generated the plan must be started from Neptus, where the desired control configuration is added to the path.

A loiter manoeuvre can be included in the landing plan before the start of the path along the virtual runway. The loiter manoeuvre can be used to control when the UAV should start it's final approach, which is practical when performing a dynamical landing.

5.2 Navigation system

The navigation system is control by a state machine 4.2.1, which is used to control the content of the output IMC messages EstimatedState and NavSources. Depending on which state the navigation system is in the IMC EstimatedState message will either have position solution form the RTK-GPS system or the external navigation system. During a short loss of the RTK the external navigation position is compensated with the average difference between the RTK solution and the external navigation solution.

5.2.1 RTK-GPS system

The RTK-GPS solution is dispatched from the DUNE task RTKGPS, however before the message is accepted by the Navigation task the message must include a valid base station position. The base station position is not included in the output message from RTKlib, which demand the base station position to be calculated locally at the base station as a standalone GNSS receiver. For this purpose the DUNE task BasestationFix is used to lock the current position of the base station, which result in the base station position being transmitted to the RTKGPS task. The navigation system require to now the reference position of the base station in order to use the RTK-GPS solution. However the base station position is currently not part of the output message from rtkrcv. This is resolved by allowing the base station to calculate it's own position as a standalone GPS. The GPS position is transmitted to a local Dune task on the base station, where the operator can decide when the base station can be considered as fixed. When the base station is considered fixed the position is sent to the X8, where it's included in the RTK-GPS solution message.

Nest system

A nest system is a stationary unit with the sole purpose of providing it's position to the rest of the Dune System. As part of the navigation system the base station is defined as a nest, where the GPS position is sent to the RTK-GPS system when fixed. An other nest has been created to obtain the GPS position of the stationary net. The net nest is configured as a rover in RTK-GPS configuration, such that the position relative to the base station is in the same frame as the X8.

5.2.2 Operator interface

The state of the navigation system is monitored though a interface in Neptus. The interface indicate which source the Dune system is using for state information. The interfaced apply a color code to indicate which source is currently in use in addition to all sensor system that are available, as seen in table 5.1.

Color	**Description**
White	Not available
Yellow	Available, but not in use
Green	Available, and in use

Table 5.1: Net approach parameters

Chapter 6

SIL test

The system was tested through a Software In the Loop (SIL) simulation. In a SIL simulation the same code that runs in the hardware is tested against a simulator which behave similarly to the real system.

6.1 Setup

When performing a SIL test the Dune system is locally connected to ardupilot, which is communicating with JSBsim instead of the real UAV. The simulation is performed with a mathematical model of the X8 developed in the Master thesis [ref master thesis]. The purpose with the SIL test is to test the same software which will run in the X8 during a experimental test, and is used to verified that the system works. However since it's a simulated environment the results can only be considered as ideal results.

6.2 Path planing SIL

The landing path is tested in both guided and FBWA.

6.3 Navigation SIL

The navigation system was tested by creating a RTK-GPS simulator. This allowed for testing of the navigation interface, including the fixing of the base station position.

6.3.1 Short rtk loss compensator

The short rtk loss compensator was tested by adding a bias in the position solution from the simulated GpsFixRtk message.

Chapter 7

Experiment

7.1 Path

7.2 Navigation

7.2.1 Short loss compensator

Include result when short loss was not in use. Compare to result when it's in use.

References

Ubiquiti networks rocket m powerful 2x2 mimo airmax basestation models: M5, rm5-ti, m3, m365, m2, rm2-ti, m900 datasheet. https://dl.ubnt.com/datasheets/rocketm/RocketM_DS.pdf. Accessed: 16.05.2016.

Brian A Barsky and Tony D DeRose. Geometric continuity of parametric curves: three equivalent characterizations. *IEEE Computer Graphics and Applications*, (6):60–68, 1989.

Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

Jon S Berndt. Jsbsim: An open source flight dynamics model. In *in C++. AIAA*. Citeseer, 2004.

Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

Marcus Frølich. Automatic ship landing system for fixed-wing uav. Master's thesis, Norwegian University of Science and Technology, NTNU, 2015.

Insitu. Skyhook universal productcard. https://insitu.com/images/uploads/pdfs/Skyhook_Universal_ProductCard_PR041615.pdf. Accessed: 13.05.2016.

H Jin Kim, Mingu Kim, Hyon Lim, Chulwoo Park, Seungho Yoon, Daewon Lee, Hyunjin Choi, Gyeongtaek Oh, Jongho Park, and Youdan Kim. Fully autonomous vision-based net-recovery landing system for a fixed-wing uav. *Mechatronics, IEEE/ASME Transactions on*, 18(4):1320–1333, 2013.

Ricardo Martins, Paulo Sousa Dias, Eduardo RB Marques, José Pinto, Joao B Sousa, and Femando L Pereira. Imc: A communication protocol for networked vehicles and sensors. In *Oceans 2009-Europe*, pages 1–6. IEEE, 2009.

Maxtena. M1227hct-a-sma l1/l2 gps-glonass active antenna, data sheet. http://www.farnell.com/datasheets/1681376.pdf. Accessed: 18.12.2015.

Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements and Performance Second Edition*. Lincoln, MA: Ganga-Jamuna Press, 2011.

Novatel. Gps-701-gg and gps-702-gg, data sheet. www.novatel.com/assets/Documents/Papers/GPS701_702GG.pdf. Accessed: 18.12.2015.

Joel Pinto, Paulo S Dias, Rui P Martins, Joao Fortuna, Eduardo Marques, and Joao Sousa. The lsts toolchain for networked vehicle systems. In *OCEANS-Bergen, 2013 MTS/IEEE*, pages 1–9. IEEE, 2013.

RTKLIB. Rtklib ver. 2.4.2 manual. http://www.rtklib.com/prog/manual_2.4.2.pdf. Accessed: 18.12.2015.

Robert Skulstad and Christoffer Lie Syversen. Low-cost instrumentation system for recovery of fixed-wing uav in a net. Master's thesis, Norwegian University of Science and Technology, NTNU, 2014.

Robert Skulstad, Christoffer Lie Syversen, Mariann Merz, Nadezda Sokolova, Thor I Fossen, and Tor A Johansen. Net recovery of uav with single-frequency rtk gps. In *Aerospace Conference, 2015 IEEE*, pages 1–10. IEEE, 2015.

Tomoji Takasu and Akio Yasuda. Development of the low-cost rtk-gps receiver with an open source program package rtklib. In *international symposium on GPS/GNSS*, pages 4–6. International Convention Centre Jeju, Korea, 2009.

Peter JG Teunissen. A new method for fast carrier phase ambiguity estimation. In *Position Location and Navigation Symposium, 1994., IEEE*, pages 562–573. IEEE, 1994.

Peter JG Teunissen. The least-squares ambiguity decorrelation adjustment: a method for fast gps integer ambiguity estimation. *Journal of Geodesy*, 70(1-2): 65–82, 1995.

Antonios Tsourdos, Brian White, and Madhavan Shanmugavel. *Cooperative path planning of unmanned aerial vehicles*, volume 32. John Wiley & Sons, 2010.

U-blox. Neo/lea-m8t u-blox m8 concurrent gnss timing modules, data sheet. https://www.u-blox.com/sites/default/files/NEO-LEA-M8T_DataSheet_(UBX-14006196).pdf, a. Accessed: 18.12.2015.

U-blox. U-blox m8 receiver description including protocol specification. https://www.u-blox.com/sites/default/files/products/documents/u-bloxM8_ReceiverDescrProtSpec_%28UBX-13003221%29_Public.pdf?utm_source=en%2Fimages%2Fdownloads%2FProduct_Docs%2Fu-bloxM8_ReceiverDescriptionProtocolSpec_%28UBX-13003221%29_Public.pdf, b. Accessed: 18.12.2015.

Bjørnar Vik. Integrated satellite and inertial navigation systems. *Department of Engineering Cybernetics, NTNU*, 2014.

Artur Zolich, Tor Arne Johansen, Krzysztof Cisek, and Kristian Klausen. Unmanned aerial system architecture for maritime missions. design & hardware description. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, pages 342–350. IEEE, 2015.