NTNU – Trondheim
Norwegian University of
Science and Technology

Precision Airdrop from a Fixed-Wing Unmanned Aerial Vehicle

Simen Fuglaas

Master of Science in Cybernetics and Robotics
Submission date: June 2014
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Wireless sensor networks sometimes require that beacons are deployed at quite accurate locations. An example is deployment of tracking devices for monitoring of drifting sea ice and icebergs in the Arctic in order to support marine operations and ship traffic.

This task consist of the following steps:

1. Given a known end position for the sensor, calculate the optimal position and course of the UAV at the sensor deployment point. Assume that wind speed and direction is known, UAV speed and altitude are given.
2. Develop an algorithm that is used to control the UAV in an optimal way to the optimal deployment point.
3. Implement a system, based on the Piccolo SL Autopilot in a Penguin B UAV. Consider the use of both heading turn rates and waypoints as the primary control input. For navigation, the sensors in the Piccolo SL Autopilot should be used. Implement also an interface to a simple sensor deployment mechanism that is integrated as a payload on the Penguin B UAV.
4. Test the system using the hardware-in-the-loop simulation setup in the laboratory, and analyze the performance of the system.
5. Test the system with field experiments.

Supervisor: Professor Tor Arne Johansen

Co-supervisor: PhD Candidate Mariann Merz

Collaborator: MSc Candidate Siri Holthe Mathisen

Preface

This thesis concludes my five year long journey towards a degree in Master of Science at the Department of Engineering Cybernetics, NTNU. The contents of this thesis does not express the many hours spent programming and debugging code. Nor does it reflect the joy of success and the sadness and frustration that followed a failure. Nevertheless, writing this thesis has been the largest project thus far in my life and it has been an educational experience which I will remember for the rest of my days.

This thesis could, however, not be completed without the help and support of many. I would like to thank my supervisors, Tor Arne Johansen and Mariann Merz, for guidance and support throughout the duration of the project. Further I would like to thank Øyvind Ulvin Halvorsen, Erlend Jørgensen, Kristian Stormo, Håkon Bøe, Håkon Søhoel and Sverre Kvamme, with whom I shared both office and many good times. I would also like to thank my collaborator, Siri Holthe Mathisen, for many constructive discussions and for putting up with me. Lastly I would like to thank the reader whom thus far have considered reading my thesis, and I do hope that you have been inspired to continue.

Thank you!

Simen Fuglaas

Trondheim, June 2014

Abstract

Accurate mapping of the polar regions requires reliable placement of wireless transmitting sensors, also known as beacons, on icebergs and drift ice. This thesis considers the use of a specific fixed-wing unmanned aerial vehicle, known as the Penguin B, to accurately deploy said beacons from the air. An analysis of the possible precision airdrop methods was conducted and the decision was made to release the beacon in free fall from the aircraft. The estimated trajectory was calculated and used to decide the optimal release position and direction. Combined this is known as the release configuration. Moreover, two different aircraft path planning algorithms were developed in order to achieve the desired configuration.

The final system, including the necessary hardware and software, was implemented into the provided framework. This system was further tested through simulations in the laboratory in addition to some field testing. The simulations revealed that with the most advanced path planning algorithm, it was possible to achieve a close to optimal release configuration. This further resulted in an airdrop where the accuracy of the impact depended primarily on the altitude of release, in addition to the unpredictable environmental factors, such as wind gusts. The field tests displayed that the system was successfully implemented into the provided framework. However, unforeseen technical difficulties related to the aircraft, outside the control of this project, prevented in-air testing.

Under the assumptions made throughout this thesis, the simulations revealed that the implemented system was able to reliably deploy the beacon such that it landed within a relatively small perimeter around the target.

Sammendrag

Nøyaktig kartlegging i polarområdene krever pålitelig plassering av sensorer med radiosendere på isfjell og drivis. Denne oppgaven betrakter bruken av et ubemannet fly, av typen Penguin B, til nøyaktig utplassering av slike sensorer fra luften. En analyse av mulige metoder for å gjennomføre et presisjons-slipp fra luften ble gjennomført, og den valgte løsningen var å slippe sensoren i fritt fall fra flyet. Sensorens estimerte bane ble beregnet og brukt til å bestemme den optimale slipp-posisjonen og slipp-retningen, samlet definert som slipp-konfigurasjonen. Videre ble det utviklet to forskjellige baneplanleggings algoritmer for det ubemannede flyet slik at det ville oppnå den ønskede konfigurasjonen.

Det endelige systemet, inkludert nødvendig maskinvare og programvare, ble implementert og integrert i det ubemannede flyet. Dette totalsystemet ble ytterligere testet gjennom simuleringer i laboratoriet, i tillegg til noe felttesting. Simuleringene viste at, med bruk av den mest avanserte bane-planleggings algoritmen, var det mulig å oppnå en nær optimal slipp-konfigurasjon. Dette resulterte videre i et slipp fra luften, der nøyaktigheten av treffet først og fremst var avhengig av flyets høyde ved slippet, samt uforutsette forstyrrelser som f.eks. vind. Felttester på bakken viste at totalsystemet fungerte som forutsatt. Uforutsette tekniske problemer med flyet, utenfor prosjektets kontroll, hindret, imidlertid, testing av systemet i luften.

Med de forutsetningene og antagelsene som er gjort i oppgaven, viste simuleringene at det utviklede og implementerte totalsystemet var i stand til å utplassere sensoren med radiosender relativt nøyaktig.

Contents

1. Introduction

1.1. About the Project

This project will consider the use of a fixed-wing Unmanned Aerial Vehicle (UAV) in order to deploy wireless transmitting sensors, also known as beacons, accurately on a given target. This target may reside far from the ground station and it may possibly be located in an uninhabited area. As such, landing the aircraft is often not an option. For that reason, it was desirable to develop a way to accurately and precisely deploy the beacons from the aircraft.

1.1.1. Motivation

In the Arctic regions there are several thousand icebergs which may pose a threat to both the petroleum industry and the ship traffic. The largest icebergs can be detected by radar satellite imagery, however, that is not always the case with the smaller icebergs [1]. Consequently, a more accurate tracking by the use of deployed beacons may prove to be quite useful in mapping the Arctic regions.

There are several benefits of using a fixed-wing UAV to perform this task. Firstly, the use of a fixed-wing aircraft offers a significantly longer range than a multicopter. Since operations over the ocean

often require a long range vehicle, endurance was chosen over the ability to hover and land. Secondly, it is far cheaper to utilize small UAVs rather than deploying the beacons from a full-scale helicopter (or airplane) or from a ship.

It is also worth mentioning that the center for Autonomous Marine Operations and Systems (AMOS) at the Norwegian University of Science and Technology (NTNU) desires, among others, to develop technology to support mapping and monitoring operations in the remote regions of the Arctic. Consequently, the results of this thesis may possibly be used as a part of this operation.

1.1.2. Core Assumptions

This project was restricted to the use of a designated fixed-wing aircraft system. Consequently, a couple of assumptions and constraints, related to this system, were specified as a part of the final solution.

First, the aircraft would operate no lower than 30 m above ground level. However, it was also possible that the aircraft would have to fly at a higher altitude.

Second, upon choosing an operational altitude, it was, for simplicity, also assumed that this altitude would remain constant throughout the precision airdrop operation. This implied that the control of the vehicle was restricted to the horizontal plane.

Another assumption which had to be considered was that the aircraft would cruise at a constant speed, namely 28 m/s.

Furthermore, this thesis will primarily consider the initial point of impact of the beacon, and as such disregard any possible displacement that may occur after the impact.

1.1.3. Approach

The problem was approached by first considering the various possible ways of accomplishing a precision airdrop from a fixed-wing aircraft. Further it was necessary to decide which method would be the most ideal and could provide a sufficiently high degree of accuracy and reliability. It was also necessary to develop an algorithm that would control the aircraft throughout the operation.

This system was then implemented into the provided framework. This task included assembling a payload with the needed hardware to perform computations and also integrate the precision airdrop solution onto the plane. Additionally, it was necessary to develop the required software, which would perform the necessary computations and provide the autopilot with the control commands.

Simulations were performed both throughout the development of the system, and after completing the final system, in order to determine the achieved performance. Moreover, some field testing was conducted, but due to unforeseen issues with the aircraft, these tests were of limited value.

1.1.4. Distribution of Work

As mentioned in the preface, the project on which this thesis was based, was conducted in cooperation with MSc Candidate Siri Holthe Mathisen. Thus the tasks were divided between Siri Mathisen and the author in the following way.

Siri Mathisen and the author would both research and determine a desirable method of precision airdrop. Upon choosing this method,

the author would perform the necessary calculations and physical assumptions in order to find the optimal position and course of the UAV at the time of deployment. Furthermore, the author researched some possible control strategies and, along with Siri Mathisen, determined which solution was the most suitable for this operation.

The implementation of the system was in general conducted together. However, the author focused primarily on the software implementation of finding the optimal position and course, along with the development and implementation of the path planning and path tracking algorithms. Siri Mathisen was, on the other hand, responsible for the hardware architecture of the payload, which included both the on board computer as well as the hardware required by the airdrop mechanism. Siri Mathisen would also set up the communication link between the payload and the ground unit and develop the necessary software to activate the airdrop mechanism.

Siri Mathisen and the author both participated in the simulations at the UAV laboratory as well as the field testing which was conducted at Agdenes Airstrip.

1.2. Related Work

Deployment of objects from different fixed-wing aircrafts has previously been studied extensively for the valuable military purpose it serves. Ducote and Speelman [2] explained that delivery by air has existed since the early days of aviation history. However, it was not until after World War II that the need for precision airdrop expanded. As the armed forces were capable of moving further and faster, the need for fast resupply became a problem.

Many different strategies were considered in order to safely deliver the airdrops. In some cases the geographical topography allowed for low-altitude flight, and as such, better accuracy was achieved. Other cases demanded an airdrop from higher altitudes in order to avoid radar detection or ground-to-air artillery [2]. Over the years, a large number of different techniques were developed to accomplish this mission. However, in many cases the mass of the released object has been of such magnitude that a larger aircraft was a necessity.

In recent times, the purpose of a precision airdrop has expanded to also include civilian use, such as the one presented in this thesis. Operations in ice-regions are often very expensive due to their remote location [1]. This cost can be reduced significantly by utilizing precision airdrop from a UAV. Tiffin et al. [1] explained that a precision airdrop from a UAV has been tested on Antarctic icebergs, but there are, however, no ready-made systems available.

1.3. Outline of the Thesis

This thesis comprises the necessary steps and decisions which were made throughout this project along with an analysis of the resulting system. Chapter 2 will present the provided system and framework around which this thesis was built. In chapter 3 an analysis of a few different proposed precision airdrop methods is conducted, before further investigating the necessary calculations in each method along with the development of the aircraft control. The payload, including the hardware and other necessary components, is introduced in chapter 4. Furthermore, this chapter presents the implemented software, along with an explanation of the code structure. Chapter 5 contains the conducted simulations along with an analysis of the results. The

performance of the system as a whole is discussed in chapter 6, and final conclusions are made in chapter 7. Lastly, chapter 8 will introduce some thoughts for possible future work related to this system.

2. System Description

The fuselage, the aircraft engine and the flight management system constitute the main framework within which this thesis was based. As such it was important to build the thesis around this system. This means that it was imperative to consider both the strengths of the system, as well as the limitations, before any decisions could be made. This chapter intends to give a brief introduction to some of the physical properties of the fixed-wing aircraft and provide the reader with some basic background knowledge of the chosen hardware and software.

2.1. Coordinate Systems

A brief introduction to the relevant coordinate systems is in order. It is common to utilize the American Global Navigation Satellite System (GNSS), known as the Global Positioning System (GPS), in order to locate vehicles on, or near, the surface of the Earth. This system provides an exceptionally good estimate of the position of the vehicle on a global scale. However, for the purpose of tracking and guiding a UAV, it is of great help to reduce the global coordinate system down to a local coordinate system.

The Earth-centered Earth-fixed (ECEF) frame is a global coordinate system where the origin is fixed in the center of the Earth. The x-axis points towards the intersection of 0° longitude (Greenwich meridian) and 0° latitude (Equator). The z-axis is parallel to the rotational axis of the Earth and the y-axis completes the right handed orthogonal coordinate system [3]. This coordinate system can be seen in Fig. 2.1 denoted as (x_e, y_e, z_e). The positions given by the GPS are most often provided as ellipsoidal coordinates (latitude, longitude and height) based on the World Geodetic System 84 (WGS84) ellipsoid.

The North-East-Down (NED) frame is commonly utilized to get a better resolution of the area of interest, locally. This frame is defined relative to the Earth's reference ellipsoid in the following way. The z-axis points downwards perpendicular to the plane tangent to the ellipsoid. The x-axis points towards true north[1] and the y-axis points to east [3]. This frame is accurate in the local area of interest, as shown in Fig. 2.1, but the origin of the NED frame needs to be reset if the vehicle moves too far from the set NED origin.

The BODY frame of the vehicle is defined to move and rotate in accordance with the vehicle. The origin of this frame is fixed to a predetermined point on the vehicle. The axes are chosen so that x points in the forward direction, y points to the right side and z points downward. Moreover, the Euler angles, as defined in Fig. 2.2, relates the BODY frame to the NED frame [3].

[1] True north, meaning the geodetic north pole, as opposed to the continuously moving magnetic north pole.

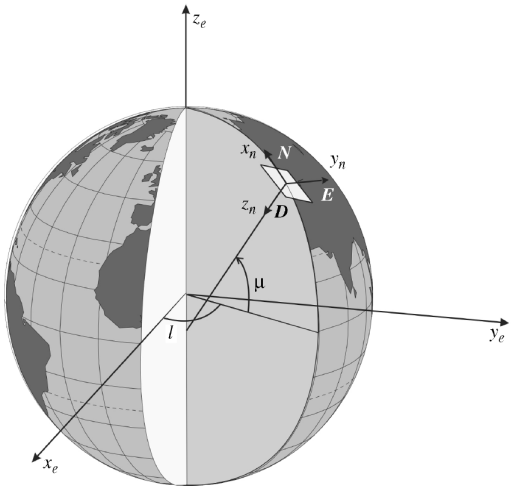


Figure 2.1.: The NED frame (x_n, y_n, z_n) shown relative to the ECEF frame (x_e, y_e, z_e). [4]

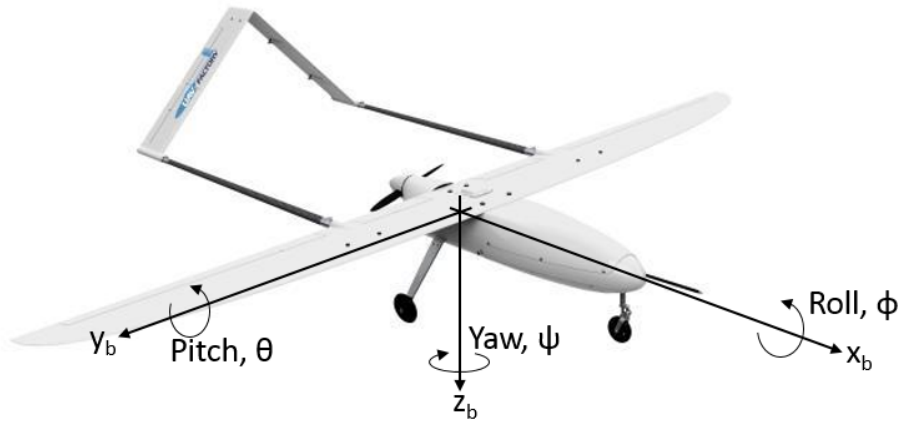


Figure 2.2.: The Penguin B with the chosen BODY frame (x_b, y_b, z_b) and the associated Euler angles (ϕ, θ, ψ).

2.2. Fixed-Wing Aircraft

The fixed-wing aircraft differs in many ways, some of which have previously been mentioned, from the multicopter. An important difference to consider is the fact that the multicopter, when hovering[2], is a holonomic system, whereas the fixed-wing aircraft is a non-holonomic system. What this means is that the multicopter can independently control all the six degrees of freedom (x, y, z, roll, pitch and yaw) while it is hovering. The fixed-wing aircraft, on the other hand, is unable to control the roll, pitch and yaw independently of its x, y and z positions respectively [5]. As such, it is more difficult to plan a trajectory for the fixed-wing aircraft.

2.2.1. Kinematic Equations of Motion

Ren and Beard [6] explained that a fixed-wing aircraft equipped with low-level altitude-hold, velocity-hold and heading-hold autopilots can be modeled by kinematic equations of motion that are similar to those of a non-holonomic mobile robot. As previously stated in sec. 1.1.2, the altitude is assumed to be constant throughout the precision air-drop mission. As such, modeling the system as a vehicle with three degrees of freedom, as done by Singh and Fuller [7], by the use of the kinematic equations of motion is a good approximation.

The states to be modeled are the position (x, y) the speed v and the heading angle ψ.

[2]The multicopter is a non-holonomic system when it is not hovering.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ v \\ \psi \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{2.1}$$

The kinematic equations of motion can then be modeled as seen in equation 2.2 [7].

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \\ \dot{v} \\ \dot{\psi} \end{bmatrix} \tag{2.2}$$

In order to account for the dependencies in the system, it may be rewritten as

$$\dot{\mathbf{x}} = \begin{bmatrix} x_3\cos(x_4) \\ x_3\sin(x_4) \\ u_1 \\ u_2 \end{bmatrix} \tag{2.3}$$

where $u_1 = \dot{v}$ and $u_2 = \dot{\psi}$ represent the actuators of the vehicle system. However, due to the inherent properties of the fixed-wing aircraft there are a few constraints that must be considered. First, the speed of the vehicle needs to stay above a positive minimum threshold in order to stay airborne. Nor can the aircraft exceed an upper positive value due to the limitation in thrust. Moreover, a vehicle is unable to accelerate infinitely fast and consequently the acceleration must be constrained to reasonable values. Secondly, the heading turn rate, which will be revisited in sec. 2.2.2, can saturate due to the limitation in the roll

angle. [6, 7]

Consequently, the following constraints on the actuators can be posed.

$$(u_i)_{min} \leq u(t) \leq (u_i)_{max} \qquad i = \{1, 2\} \tag{2.4}$$

$$v_{min} \leq v(t) \leq v_{max} \tag{2.5}$$

However, it has also been stated that the speed of the UAV will remain constant and it is, as such, uncontrollable. Thus, the equations of motion that will be considered are reduced to

$$\dot{\mathbf{x}} = \begin{bmatrix} V_A \cos(x_4) \\ V_A \sin(x_4) \\ u \end{bmatrix} \tag{2.6}$$

where V_A is the constant airspeed, and the controllable inputs have been reduced to one, namely the heading turn rate $u = \dot{\psi}$. This gives the following constraint on the system.

$$u_{min} \leq u(t) \leq u_{max} \tag{2.7}$$

2.2.2. Bank and Turn

In order for a fixed-wing aircraft to alter its heading it is necessary to perform a banked turn, i.e. change its roll Φ as illustrated in Fig. 2.3. Leven et al. [8] suggested that all turn maneuvers can be considered as coordinated turns. By this term they implied that all turns are executed at a constant flight speed and that the sum of the forces along the lateral axis of the aircraft equals zero.

Based on the aforementioned statements, the following equation is proposed [8].

$$F_C = F_L \sin(\Phi) \tag{2.8}$$

where F_C represents the centrifugal force, F_L is the lift force and Φ is the roll angle. Inserting for these values according to Fig. 2.3 gives the following expression for the heading turn rate $\dot{\psi}$ with respect to the roll Φ.

$$m\dot{\psi}V_A = mg\tan(\Phi) \tag{2.9}$$

$$\dot{\psi} = \frac{g}{V_A}\tan(\Phi) \tag{2.10}$$

Now, given a known maximum roll angle Φ_{max} it is possible to calculate the maximum heading turn rate $\dot{\psi}_{max}$ by using equation 2.10. V_A is the constant air speed and g represents the gravity acceleration.

Furthermore, with knowledge of the maximum roll angle Φ_{max} it is possible to calculate the smallest achievable turn radius R i.e. the sharpest turn possible by the UAV. In the absence of wind and sideslip, Beard and McLain [9] proposed the following expression for the turning radius of a coordinated turn.

$$R = \frac{V_A^2}{g\tan(\Phi)} \tag{2.11}$$

However, in order for the minimum turn radius R to be as accurate as possible, it is necessary to fully account for the effects of the wind. This can be done in a simple way by claiming that the maximum

achievable speed is given by the sum of the constant airspeed V_A and the magnitude of the wind $W_{magnitude}$.

$$R = \frac{(V_A + W_{magnitude})^2}{g \tan(\Phi)} \tag{2.12}$$

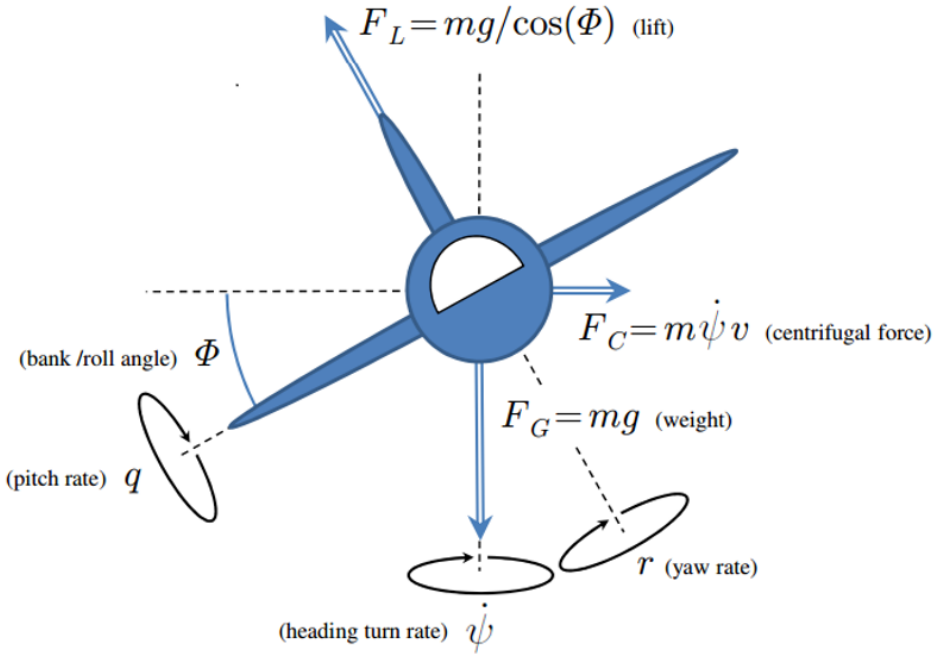


Figure 2.3.: An aircraft executing a coordinated, level turn with constant yaw rate r and pitch rate q. [8]

2.2.3. Airspeed

The definition of the constant airspeed V_A may appear somewhat vague. Thus an elaboration of this definition is appropriate. According to the Federal Aviation Administration (FAA) there are four

different kinds of airspeed, each of which will presented below as provided by the FAA [10].

Indicated Airspeed (IAS) means the speed of an aircraft as shown on its pitot-static airspeed indicator calibrated to reflect standard atmosphere adiabatic compressible flow at sea level uncorrected for airspeed system errors.

Calibrated Airspeed (CAS) means the indicated airspeed of an aircraft, corrected for position and instrument error. Calibrated airspeed is equal to true airspeed in standard atmosphere at sea level.

Equivalent Airspeed (EAS) means the calibrated airspeed of an aircraft corrected for adiabatic compressible flow for the particular altitude. Equivalent airspeed is equal to calibrated airspeed in standard atmosphere at sea level.

True Airspeed (TAS) means the airspeed of an aircraft relative to undisturbed air. True airspeed is equal to equivalent airspeed multiplied by $\frac{1}{2}\frac{\rho_0}{\rho}$.

It is desirable to obtain the true airspeed in order to get the most accurate measurement of the airspeed. However, unlike larger aircrafts, the UAV intends to fly at a relatively low altitude, which further implies that the indicated airspeed can be approximated equal to the true airspeed. Thus the following expression will be assumed valid.

$$\text{IAS} = \text{TAS} = V_A \tag{2.13}$$

2.3. Tools and Equipment

2.3.1. Penguin B

The chosen UAV platform is called Penguin B and is manufactured by UAV Factory. According to the information provided in its datasheet [11], the Penguin B is a high performance unmanned aircraft, capable of flying for 26.5 hours with 4 kg payload. This makes the aircraft ideal for flying to remote locations.

The cruise speed of the Penguin B is set to be 22 m/s, while the max level speed is 36 m/s. A lower speed can also be achieved, but for the purpose of this project it will be assumed that the cruise speed, measured as the IAS, is set to 28 m/s. It will also be assumed that the Penguin B will takeoff with the use of a Portable Pneumatic Catapult, as seen in Fig. 2.5. This enables the capability of taking off from unprepared areas. However, this may also put limitations on the placement of equipment to be used to deploy the beacons, as this equipment must not interfere with the launcher.

A closer examination of Fig. 2.4 reveals that the propeller is located behind the Penguin B, rather than in front. Although this improves the maneuverability of the aircraft, it may, however, pose a problem when deploying the beacons. It is highly undesirable to have any loose objects collide with the propeller. Consequently, any payload released from the aircraft must either have an initial velocity pushing the payload away from the aircraft body, to avoid collision with the propeller, or the payload must be released out of harms way for the propeller.

Furthermore, it is of importance to notice that the max takeoff weight may not exceed 21.5 kg. Of this amount, the empty Penguin B weighs

10 kg. This leaves room for 11.5 kg worth of payload and fuel. Naturally, this is a significant constraint to keep in mind when choosing the deployment mechanism. Moreover, with the possible need to install exterior equipment on the Penguin B, it is important to place this equipment in such a way so that it does not alter the controllability or the stability of the aircraft.

Lastly, the use of a pitot-static tube installed at the front of the Penguin B allows for measurements of the dynamic and static pressure. The interpretation of this data can provide the system with valuable data concerning the stream flow, i.e. the wind, and the indicated airspeed. This device can be seen alone in Fig. 2.6 and as installed on the Penguin B in Fig. 2.4.

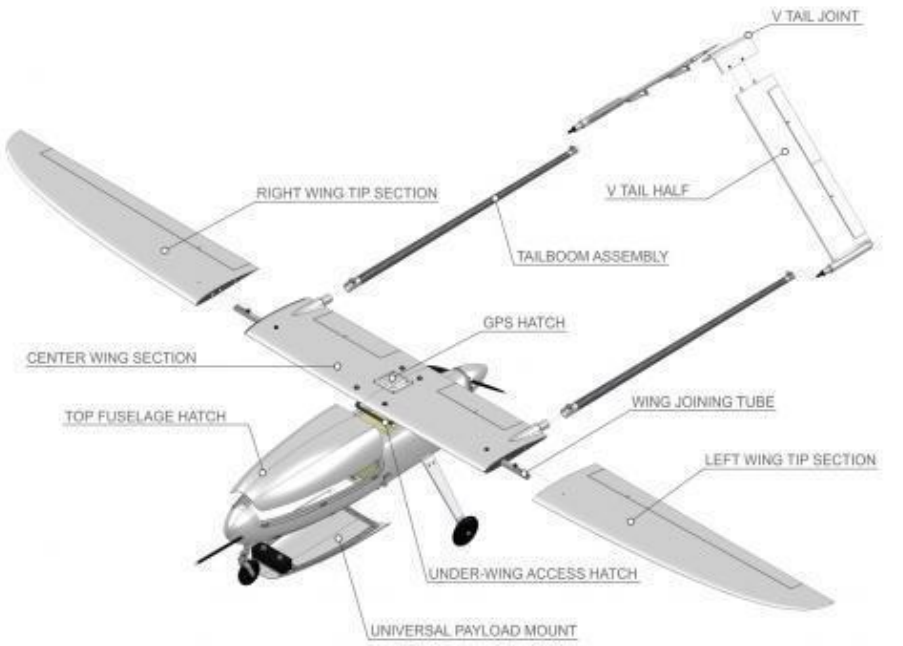


Figure 2.4.: The Penguin B platform. [12]

Figure 2.5.: Portable Pneumatic Catapult, used to launch the Penguin B. [12]

Figure 2.6.: Pitot-Static Tube. [11]

2.3.2. Piccolo SL

The chosen flight management system for the Penguin B is the Cloud Cap Technology Piccolo SL. This system offers a complete avionics solution which includes the flight control processor, inertial sensors,

ported air data sensors, a GPS receiver and a datalink radio [13]. In short terms this implies that all data concerning tracking and control must go through the Piccolo SL unit. As such, a brief introduction to this unit is appropriate.

The Piccolo SL Autopilot is responsible for interpreting the incoming flight controls and applying the appropriate actions to the flight control surfaces and the engine. The input flight control may be either waypoints or heading references, both of which will be examined more closely in sec. 3.4.

The Estimated State is comprised of the information provided by the inertial sensors, consisting of three accelerometers and three gyroscopes, and the data provided by the GPS receiver. This information ensures an accurate tracking of the position and motion at any time-instant, both in the ECEF frame and in the NED frame as well as the BODY frame.

The Estimated Stream Velocity is an approximation of the stream flow, in this case referred to as the wind, in which the vehicle traverses. As previously mentioned, the pitot-static tube installed on the Penguin B provides the static and the dynamic pressure. This data can be interpreted by the air data sensors in the Piccolo SL, which further is able to create a 3-dimensional vector describing the average wind. The reliability of this vector increases by maneuvering the aircraft in a circular pattern.

The Indicated Airspeed is, as the definition explained, based on pitot-static measurements. Consequently, the Piccolo SL air data sensors are able to calculate the indicated airspeed of the aircraft.

The Piccolo Command Center (PCC) is the command and control software of the Piccolo System. It provides a human-machine

interface (HMI) between the operator and the aircraft equipped with the Piccolo SL. Furthermore, this software makes it possible to simulate the Penguin B by loading a properly calibrated model of the Penguin B. The simulator is set up to simulate a ground station and the Penguin B such that any new control algorithms can be tested in this virtual environment. Due to the inclusion of the hardware in this simulation, it will hereafter be known as running a Hardware-in-the-Loop (HIL) simulation. It is also possible to simply run the simulation without the hardware, a so-called Software-in-the-Loop (SIL), however, this was mainly done to familiarize the author with the software.

2.3.3. Software Toolchain

The Laboratório de Sistemas e Tecnologias Subaquáticas (LSTS) located in Porto, Portugal (in English known as the Underwater Systems and Technology Laboratory), specializes in designing, constructing and operating unmanned vehicles, both below and above water [14]. In this thesis it was chosen to utilize the functionality provided by the LSTS Neptus-IMC-DUNE software toolchain. These three components will be presented below in order to give the reader an understanding of the system.

Inter-Module Communication (IMC) is a message-oriented communications protocol. These messages are placed on an IMC bus and can be used inter-process, inter-vehicle and operator-vehicle. In short terms this implies that all communication is passed by the use of this protocol. Furthermore, all of the IMC messages are created in a similar way and they are stored in a single XML file. This allows any user to easily create and add new messages

if deemed necessary. [14]

DUNE: Unified Navigation Environment is the on-board software runtime environment of the vehicle. DUNE is written in C++ and consists of several tasks which comprise the framework of the system. Each task has a specific purpose and is able to communicate with the other tasks over the IMC bus. The tasks usually run in separate threads of execution [14].

Neptus is a command and control software, written in Java, which offers a HMI for one or more unmanned vehicles. This functionality provides the operator with the possibility of planning and validating a mission before passing the executing command to the vehicle. [14]

The complete LSTS toolchain is visualized in Fig. 2.7 where it can clearly be seen how all of the communication, between Neptus and n DUNE tasks, is performed via the IMC bus. All of the available sensor data is dispatched on the IMC bus and can be extracted from any task. Some tasks may then utilize this data in a navigational algorithm and control commands are dispatched back on the IMC bus. This message is then picked up by the Piccolo DUNE task, which acts as an interpreter between the DUNE software and the Piccolo SL software, before the Piccolo SL executes the provided control commands.

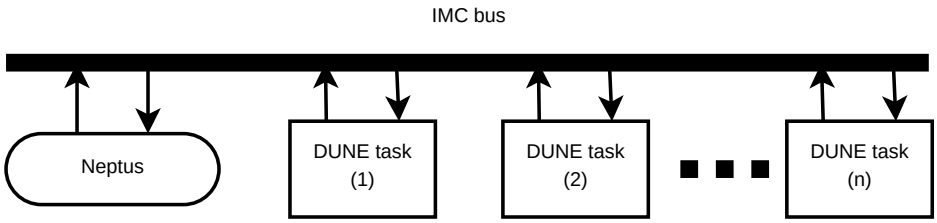


Figure 2.7.: The LSTS toolchain.

3. Analysis and Design

This chapter will begin by introducing a couple of different approaches which may achieve a precise airdrop from a fixed-wing UAV. A brief analysis of the methods, with respect to the provided framework, will be conducted and the most suitable method will be further investigated. This investigation includes a couple of important steps. Firstly, the most probable trajectory of the deployed beacon must be studied in order to obtain an understanding of where the beacon is likely to land. Secondly, a path must be generated for the aircraft to track such that the beacon deployment can be conducted according to the specifications of the beacon trajectory. Lastly, an analysis must be performed to achieve the best possible tracking of the aforementioned path.

3.1. Review of Precision Airdrop Methods

The task of accurately and precisely deploying an object from a fixed-wing aerial vehicle to a given target has been widely researched and studied because of its value to the military. In this thesis there were primarily three different approaches that were considered in order to deploy the object from the UAV – Cable-Supported Sliding Payload, Parachute Deployment and Free Fall.

3.1.1. Cable-Supported Sliding Payload

In the following method, as presented by Williams and Trivailo [15], a cable is utilized to deploy the payload. The idea suggests that the fixed-wing aircraft circles above the target point of impact with a constant radius while a cable is attached to the aircraft. The cable-tip approaches the center of this circle due to the natural dynamics of this system. Demonstrations have revealed that this method can ensure a cable-tip motion with a radius of approximately 1.5 m [16]. However, this result was obtained by utilizing a light aircraft with a 3 km long cable. Further, this result also required the cable-tip to have the correct drag-to-weight ratio.

By utilizing an anchor on the ground, as displayed in Fig. 3.1, it is possible to guarantee an accurate drop. In the case of a drop over water the cable-tip is submerged in order to provide high damping. Further studies also revealed that the descending payload needs to be slowed down in order to avoid unstable cable dynamics.

While this method provides a high degree of accuracy, it does not appear to be suitable when applied on a UAV. Relatively speaking, the sheer mass of the cable and the cable-tip object, is expected to far exceed the desired mass of the payload on the UAV. Furthermore, the task at hand desires to place the objects on uninhabited icebergs with neither anchor nor water to aid the accuracy.

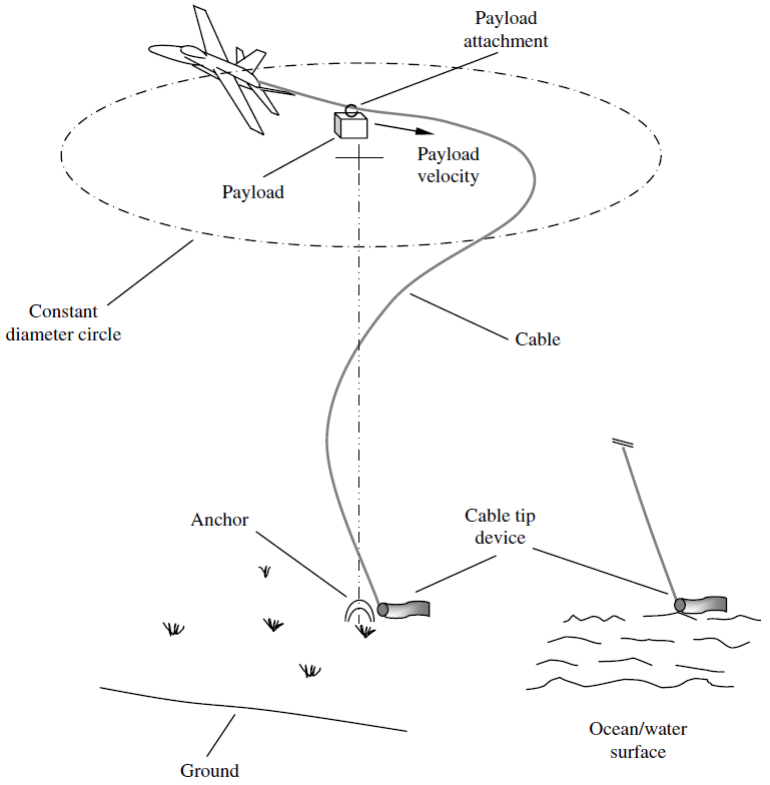


Figure 3.1.: Basic concept of cable-supported precision airdrop as visualized by Williams and Trivailo. [15]

3.1.2. Parachute Deployment

The second possible option involves releasing the payload with a parachute as to allow a softer landing. This airdrop technique has been widely studied for military purposes in order to extract larger cargo and vehicles from aircrafts [2]. Often military operations are required to adapt to the situation and the geographical location. Because of this it was desirable to develop methods of airdrop from different altitudes. In some cases it is possible for the aircraft to approach the

desired target point at a very low altitude, while in other situations the geographical location makes this approach difficult, if not impossible. Ducote and Speelman [2] separated the altitudes into the following four altitude ranges: 0 - 20 ft, 20 - X ft (where X is between 300 - 500 ft), X - 1500 ft and 1500 ft and above.

By focusing on the desirable operational altitude of the UAV it was possible to narrow down the number of possibilities. For the purpose of this mission it was most ideal to choose among the methods in the altitude range of 20-X ft. Lower altitudes would make the UAV vulnerable to changes in the terrain and high waves. Higher altitudes, on the other hand, would further complicate the system and provide less accuracy. Of the methods presented by Ducote and Speelman [2], two concepts were considered and will consequently be presented for usage with the UAV.

Parachute Low-Altitude Delivery System In this method the aircraft deploys a reefed parachute[1] as illustrated in Fig. 3.2. When the aircraft reaches the desired extraction point, the parachute is disreefed. At this point the drag forces will overcome the load restraint and the payload will be extracted. Given a known airspeed and altitude, this method of airdrop proved to provide a relatively high degree of accuracy [2]. 90 % of all the payloads landed within 25 ft of the desired point of impact. It should, however, be mentioned that this was designed for containers capable of holding 500 - 2000 lbs (approximately 226 - 907 kg).

[1] A reefed parachute means it is not fully opened. This is done to reduce the drag force on the parachute.

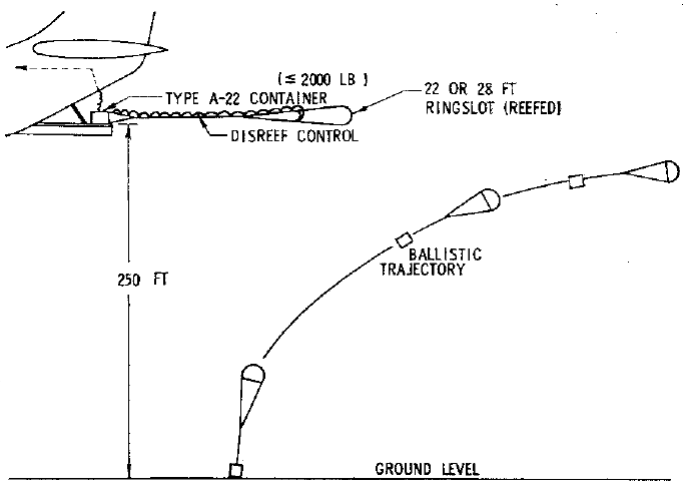


Figure 3.2.: Concept of the Parachute Low-Altitude Delivery System. [2]

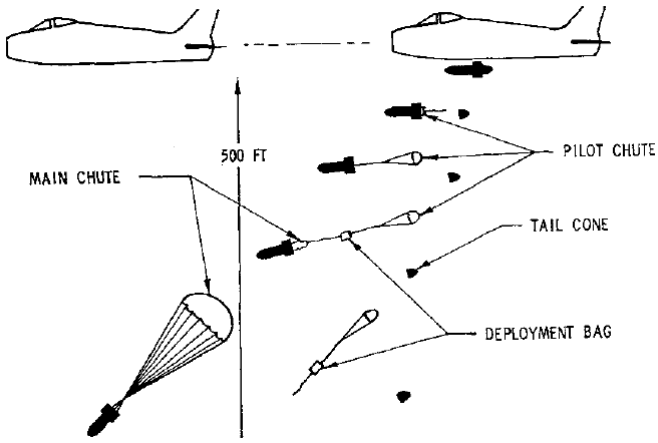


Figure 3.3.: Concept of the High-Speed Container Delivery System. [2]

High-Speed Container Delivery System As visualized in Fig. 3.3, this method releases the payload at the point of extraction. Upon release of the payload a pilot parachute is deployed from the tail of the payload. This parachute further extracts the main parachute which decelerates the payload before impact. [2]

In this mission the released payload has a far lower mass than the payloads considered by Ducote and Speelman [2]. As such, the payload is much more susceptible to be affected by wind gusts, and even more so if it is equipped with a parachute. Although the presented concepts are viable options, it is assumed that the light weight of the payload combined with a parachute would result in an overall decreased accuracy and precision.

3.1.3. Free Fall

Finally, the method of releasing the payload from the UAV without any form of aid was considered. For this method it was necessary to make a couple of assumptions. It is important that the released payload must be able to withstand a high-speed impact to the ground. The speed on impact is largely dependent on the altitude at which the payload was released. This further leads to the second assumption being that the payload can not be released from a too high altitude. This is especially important when it comes to determining the accuracy of the impact as it will almost definitely decrease with an increased altitude.

Two possibilities were considered in order to release the payload in free fall from the aircraft. One way would be to release the payload with an initial velocity perpendicular to the direction of flight. This way the payload would quickly be separated from the aircraft and would

not be of potential harm to the aircraft after the separation. Ideally this initial velocity would be in the downward direction as this would also aid in increasing the accuracy. The second way to release the payload would be without any forced initial velocity. In other words, the payload would only possess an initial velocity in the direction of flight.

3.1.4. Concluding Remarks on the Precision Airdrop Methods

A number of different approaches were considered in order to achieve a high accuracy release of the payload. It should, however, be mentioned that many of these methods were initially designed for larger objects dropped from larger planes. Naturally, it was necessary to fit the methods to a smaller scale. Furthermore, the shape of the Penguin B, and especially the placement of the propeller, provides a significant limitation when it comes to choosing the release method. A parachute may get entangled in the propeller, which ultimately can cause the plane to crash.

As for the cable-supported method, this one is very interesting and may be used with a light and thin, yet strong, cable. However, as the payload needs to be slowed during the descent, this method may prove to be difficult to realize. Additionally, the cable may also get entangled in the propeller.

In conclusion, the method of releasing the payload in free fall was considered the most suitable based on the requirements of this mission. By releasing the object safely from the aircraft (avoiding impact with the airframe structure), this method was likely to achieve an accuracy based primarily on the altitude at which it was released and any un-

predictable environmental disturbances. Consequently, this method is further investigated in this thesis.

3.2. Computed Air Release Point

The technique of releasing the object in free fall requires the calculation of a point known as the Computed Air Release Point (CARP) [17]. This is the point at which the released payload must be separated from the aircraft in order to land in the designated area of impact. The CARP needs to account for the wind, the system ballistics and the airspeed in order to be as accurate as possible. This section intends to present the calculations of this point as well as the possible sources of error involved. Moreover, two different approaches to confirm that the aircraft has reached the CARP will also be introduced.

3.2.1. Winds

When releasing the payload in free fall, without any way of controlling it after the separation from the aircraft, the wind is naturally of significant importance when it comes to the accuracy and precision of the method. The wind affects both the direction of travel in addition to the time of the fall [17]. Moreover, the wind can be highly unpredictable and can vary greatly depending on the altitude above ground level. This makes a payload release from high altitudes more unpredictable than releasing the payload from a lower altitude.

On the Penguin B the wind is measured as explained in sec. 2.3.2. This measurement, however, only describes the wind at the current altitude of the aircraft. Additionally, this measurement is simply a

mean vector of the wind profile. Regardless, as Wuest and Benney [17] explained, the CARP is often calculated by assuming a uniform wind profile from the altitude of release and to the ground. This simplification will almost certainly be a source of error which increases with an increasing release altitude. Consequently, the magnitude of this error will be further investigated in sec. 5.1.3.

In order to reduce the size of this error the choice was made to release the payload against the direction of the wind. A couple of reasons, which will now be briefly mentioned, led to this decision. First, the computation of the CARP requires a given direction of flight for the aircraft such that the released payload possesses a predictable initial velocity upon release.

Second, because the Penguin B is a relatively light aircraft, it is very susceptible to wind disturbances, such as wind gusts. Consequently, the aircraft would have to compensate for the wind by either banking or using the rudder or a combination of these. This could further lead to an undesirable aircraft configuration upon approaching the CARP, which ultimately may cause the actual point of impact to deviate significantly from the desired point of impact. Such a disturbance can be avoided, or at least mitigated, by allowing the airplane to approach the CARP against the wind direction.

Lastly, because the speed of the aircraft is reduced when flying against the wind, this may aid in achieving a more accurate release. This is a matter that will be revisited in sec. 3.2.3.

3.2.2. System Ballistics

As a free fall method has been chosen, the system ballistics, or projectile motion, will simply depend on the physical properties of the

released payload in addition to the properties of the medium that it moves through. A natural simplification here is to assume that the released payload is a smooth sphere without any means of changing the direction of flight. This further implies that the only forces acting on the released payload are the drag force and the gravitational force.

Although easily changed at a later point in time, the air density was chosen, according to Tab. 3.1 to be $\rho = 1.269\,\mathrm{kg/m^3}$. Moreover, the smooth sphere possesses a drag coefficient $C_D = 0.5$ [18], along with an initially set radius $r_{sphere} = 0.05\,\mathrm{m}$ and mass $m_{sphere} = 0.2\,\mathrm{kg}$. These are all values that can later be fine-tuned to accommodate the physical properties of the released object.

Temperature, Celsius	Air Density, $\mathrm{kg/m^3}$
+10	1.246
+5	1.269
0	1.292
-10	1.341

Table 3.1.: Density of air at Standard Atmospheric Pressure. [18]

3.2.3. Airspeed

The IAS of the aircraft has previously been assumed to remain constant throughout the precision drop maneuver as stated in sec. 1.1.2. Thus, the main inaccuracy provided by the airspeed in the free fall method comes from the rate at which sensor readings are made. If the aircraft flies at a lower speed, a higher resolution of the sensor readings can be achieved. On the other hand, with a higher airspeed, the distance between each reading increases and the resolution consequently decreases. As mentioned before, flying against the wind will

decrease the speed relative to the ground, which improves the resolution of sensor readings. Nevertheless, the airspeed, along with the possible limitation in sensor readings, may still prove to be another source of error which may decrease the overall accuracy of the airdrop.

3.2.4. Calculation of the CARP

Wuest and Benney [17] explained that the calculation of the CARP can be done by modeling the released beacon as a free fall with an initial velocity. Fig. 3.4 visualizes the concept of free fall and the active forces on the object.

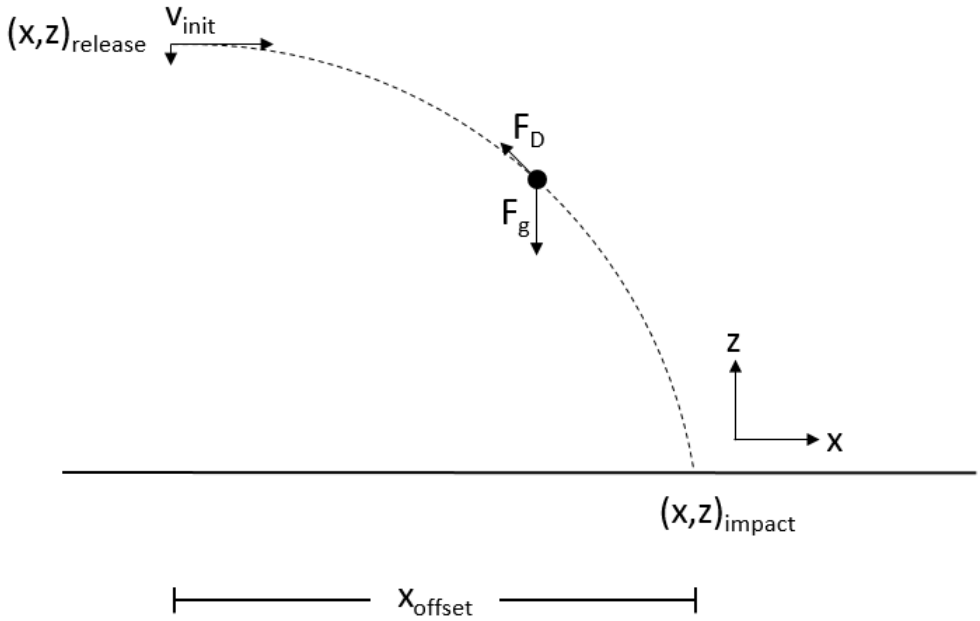


Figure 3.4.: Concept drawing of the free fall model and the active forces.

It is naturally highly desired to calculate a CARP that is as accurate as possible. Thus it is necessary to account for both the drag and

gravitational forces. Parker [19] explained that in practice the drag forces can never be neglected, nor can they be modeled linearly as this only applies to very small spheres. In light of this, the one-dimensional drag force is modeled quadratically as

$$F_D = b\bar{v}^2 \tag{3.1}$$

with

$$b = \frac{1}{2} C_D \rho A \tag{3.2}$$

Here C_D is the drag coefficient, ρ is the air density, A is the cross-sectional area and $\bar{v}$ is the velocity of the object relative to the velocity of the fluid it moves through.

Further, the motion of the projectile can be decomposed into the horizontal and the vertical direction. Expansion of this system into the third dimension is trivial.

$$ma_x = \Sigma F_x \tag{3.3}$$

$$ma_z = \Sigma F_z \tag{3.4}$$

As has been previously stated, the forces that need to be accounted for are the drag F_D and the gravitational F_g forces. Additionally, since the acceleration is time-dependent, equations 3.3 and 3.4 can be expressed as

$$m\frac{dv_x}{dt} = -F_D \tag{3.5}$$

$$m\frac{dv_z}{dt} = -F_D - F_g \tag{3.6}$$

By combining equation 3.1 with equations 3.5 and 3.6, the following expressions can be derived [19].

$$\frac{dv_x}{dt} = -\frac{b}{m}\overline{v}_x \parallel \overline{\mathbf{v}} \parallel \tag{3.7}$$

$$\frac{dv_z}{dt} = -\frac{b}{m}\overline{v}_z \parallel \overline{\mathbf{v}} \parallel -g \tag{3.8}$$

Here the bar refers to relative velocity, g is the gravitational constant on Earth and $\parallel \cdot \parallel$ is the Euclidean norm. The relative velocities can be found by accounting for the velocity of the fluid, which in this case is referred to as the wind $\mathbf{w}$.

$$\frac{dv_x}{dt} = -\frac{b}{m}(v_x - w_x) \parallel \mathbf{v} - \mathbf{w} \parallel \tag{3.9}$$

$$\frac{dv_z}{dt} = -\frac{b}{m}(v_z - w_z) \parallel \mathbf{v} - \mathbf{w} \parallel -g \tag{3.10}$$

In modeling the free fall it was desired to calculate where the object would land. In other words, it was not sufficient to just calculate the velocity at the time of impact, but also compute the position at the time of impact. This can easily be accomplished by using that the time-derivative of the position equals the velocity. As such, the system of ordinary differential equations (ODEs) that must be solved becomes

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -\frac{b}{m}(v_x - w_x) \parallel \mathbf{v} - \mathbf{w} \parallel \\ -\frac{b}{m}(v_y - w_y) \parallel \mathbf{v} - \mathbf{w} \parallel \\ -\frac{b}{m}(v_z - w_z) \parallel \mathbf{v} - \mathbf{w} \parallel -g \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{3.11}$$

However, solving this system to obtain the position of impact is not sufficient as it does not directly deliver the CARP. This was solved by setting the tentative point of release $\overline{(x, y)}_{release}$ to equal the desired point of impact $(x, y)_{impact}$. Further, $z_{release}$ was set to equal the altitude of the UAV at the time of release. By solving the system of ODEs in equation 3.11 with $(x, y)_{impact}$ as the initial position it was possible to obtain a tentative position of impact $\overline{(x, y)}_{impact}$.

As shown in Fig. 3.5, $(x, y)_{release}$, i.e. the CARP, was found by simply shifting the points. The resulting expressions can be seen in equations 3.14 and 3.15.

$$\Delta x = \overline{x}_{impact} - x_{impact} \tag{3.12}$$

$$\Delta y = \overline{y}_{impact} - y_{impact} \tag{3.13}$$

$$x_{release} = x_{impact} - \Delta x \tag{3.14}$$

$$y_{release} = y_{impact} - \Delta y \tag{3.15}$$

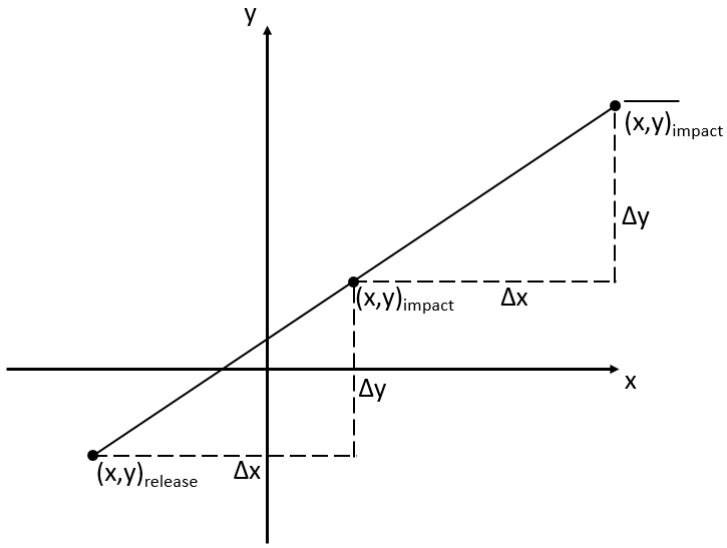


Figure 3.5.: Finding the CARP $(x, y)_{release}$ by using the tentative position of impact $\overline{(x, y)}_{impact}$.

3.2.5. Release Criteria

Although it is an intuitive observation, it is important to note that the CARP is defined as a single point in space. However, due to inaccuracies in measurements, in addition to the unexpected disturbances that occur in the real world, it is very unlikely that the UAV will ever be at this exact point. Because of this it is imperative to find a criterion which ensures that the UAV detects that it is sufficiently close to the desired release point. Furthermore, as previously mentioned, the altitude will not be controlled in this thesis and will as such also be omitted in the analysis of the release criteria.

3.2.5.1. Circle

The immediate solution, which may appear to be satisfactory, is to check the distance from the current position to the CARP. If this distance b is sufficiently small, the flyby will be considered successful, as seen in the equation 3.16 [9]. Here $\mathbf{p} \in \mathbb{R}^2$ represents the position at any time instant t and $\mathbf{z} \in \mathbb{R}^2$ equals the CARP.

$$\| \mathbf{p}(t) - \mathbf{z} \| \leq b \tag{3.16}$$

This method is, however, prone to tracking errors and requires frequent position updates. Additionally, sudden wind gusts may cause deviation from the desired path. Consequently, if b is chosen too small, the release criterion may never be met. On the other hand, with an increasing b, the accuracy of the criterion decreases. As such the size of b will have to be tuned appropriately. In the case of deploying the beacon accurately, it may be desirable to only consider the flyby a success when the UAV is actually within a certain perimeter.

This release criterion can be seen in Fig. 3.6 as the circle defined by $(x, y)_{release}$ and the radius R_{circle}.

3.2.5.2. Halfplane

The second solution to be proposed also considers the distance from the UAV to the CARP. However, in this method the release criterion is successful when a set halfplane $\mathcal{H}(\mathbf{z}, \mathbf{q})$ is crossed [9]. In equation 3.17, $\mathbf{z} \in \mathbb{R}^2$ equals the CARP and $\mathbf{q} \in \mathbb{R}^2$ is a unit vector pointing in the direction the halfplane expands into. As before, $\mathbf{p} \in \mathbb{R}^2$ is the current position of the UAV.

$$\mathcal{H}(\mathbf{z}, \mathbf{q}) \triangleq \left\{ \mathbf{p} \in \mathbb{R}^2 : (\mathbf{p} - \mathbf{z})^T \mathbf{q} \geq 0 \right\} \tag{3.17}$$

Utilizing a halfplane as a release criterion ensures a successful release regardless of wind and tracking errors. However, due to the large area in which the release criterion is valid, this method desires a control strategy that keeps the UAV as close to the desired path as possible.

The halfplane is identified in Fig. 3.6 as the area on the right of the dotted vertical line crossing through $(x, y)_{release}$.

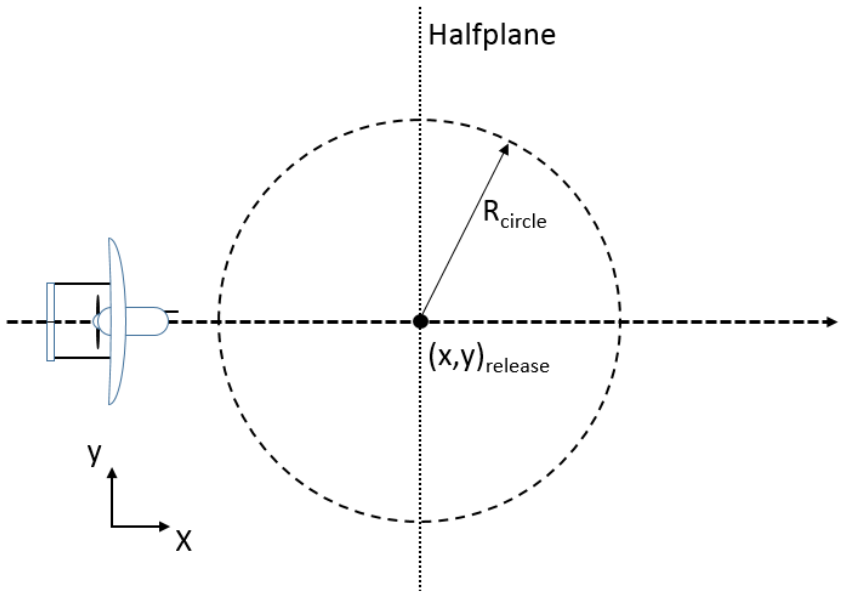


Figure 3.6.: Conceptual illustration of the difference between the two release criteria – Circle and Halfplane.

3.3. Path Planning

Thus far it has been decided to release the payload at the CARP and in the direction opposite to that of the wind in order to achieve the most optimal result. This can only be achieved by controlling the aircraft into this designated position and direction. For many cases it is possible to simply look at the transition from one position to another position, without any consideration for the orientation at either point. However, it has been presented thus far that it was necessary to consider a path which also accounted for the orientation of the vehicle at the end point. Due to the non-holonomic nature of the fixed-wing aircraft this implies that the vehicle must approach the desired position with the desired course, combined known as the desired configuration.

This section will present the different paths which were considered in order to achieve the desired end configuration.

3.3.1. Straight Line Approach

Initially, this project desired to design a simple algorithm that would easily guide the UAV into the desired configuration. Consequently, the first method would be very simple, yet accomplish the desired task, and as such achieve the desired configuration. This method attempts to guide the aircraft from any given initial configuration and towards an intermediate waypoint. This waypoint is located along a straight line which coincides with the CARP and is aligned with the desired direction the aircraft needs upon releasing the payload. This path was named the Straight Line Approach (SLA) and the idea is illustrated in Fig. 3.7. Here it can be seen that two waypoints are generated. The

final waypoint, WP_2, is located at the same position as the point of release $(x, y)_{release}$, i.e. the CARP. The first waypoint, WP_1, is chosen a distance d in the direction of the wind away from WP_2.

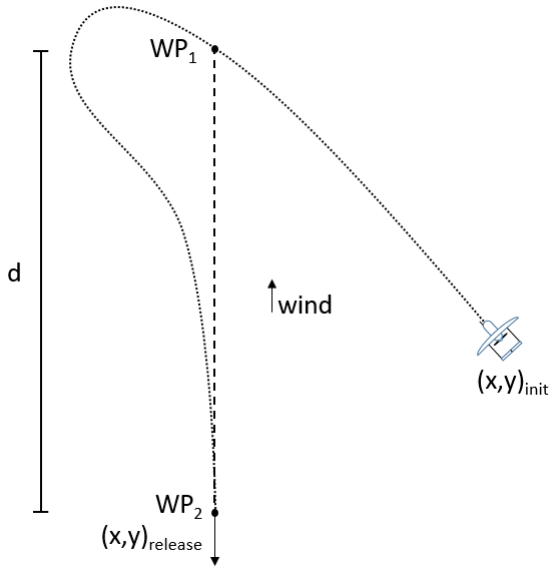


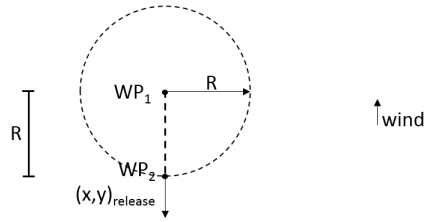
Figure 3.7.: Concept drawing of how waypoints can be used to guide the aircraft.

3.3.1.1. Choosing the Distance *d*

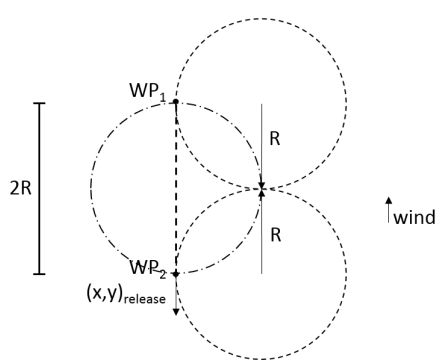
The task of choosing the optimal distance between the waypoints is not trivial as it depends on the airspeed and the heading turn rate of the aircraft. These two parameters further decide the minimum turn radius R of the aircraft, as seen in equation 2.12. Furthermore, d depends on the method used to track the path, as some methods might have a better performance than others. However, as an initial guess, it is assumed that d is set according to the constraint provided by the minimum turn radius R.

A further analysis of this distance is conducted in Fig. 3.8. Here Fig. 3.8a reveals that with $d = R$ the aircraft will only be able to fulfill the goal by entering WP_1 in the desired direction of flight with only a minor angle offset. If the aircraft enters WP_1 at any other angle it will make the sharpest turn possible and enter WP_2 at the wrong angle. In Fig. 3.8b it can be seen, by following the thick dashed line and the thin dashed line, that the aircraft can accomplish the goal by entering WP_1 from below and above. However, a quick glance at the dot-dashed line reveals that an approach from the sides does not achieve the desired configuration.

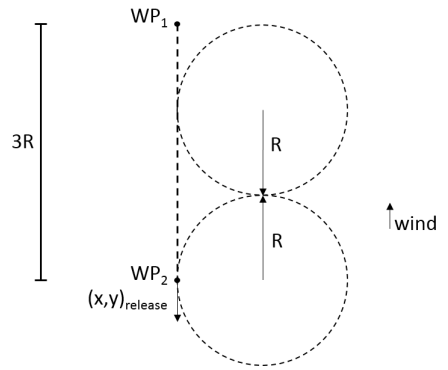
Finally, Fig. 3.8c illustrates that it should be theoretically possible to accomplish the goal by setting $d = 3R$. At this distance the aircraft should have enough space to enter WP_1 at any angle and still enter WP_2 at the desired angle. Nevertheless, for the purpose of testing, the distance is chosen to be $d = 4R$ to account for the uncertain behavior of the autopilot as well as possible wind disturbances.



(a) $d = R$



(b) $d = 2R$



(c) $d = 3R$

Figure 3.8.: A brief schematic explaining the necessity of increasing the distance d by multiplying the minimum turn radius R.

3.3.1.2. Concluding Remarks on the Straight Line Approach

It can easily be seen that this method provides a very simple approach to the problem. Regardless of the initial position and orientation, the UAV will approach WP_1 with an arbitrary angle of attack depending on the initial configuration. Moreover, with a sufficiently large d the UAV is able to straighten up before reaching WP_2. However, this method is likely to require a very large d in order to achieve a satisfactory result, which might not always be desirable.

3.3.2. Dubins Path

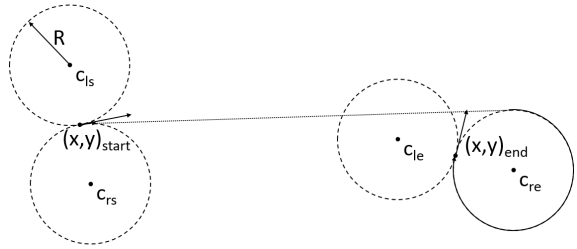
Another approach is to use a path known as Dubins path, which, in this section, will be based on the interpretation and presentation shown by Beard and McLain [9]. This method assumes an initial configuration and a terminal configuration, and further attempts to find the time-optimal path between the two points. This path consists of a circular arc, followed by a straight line and finally a second circular arc. The idea of the circular arcs is to define their radius large enough for the coordinated turn radius constraint, given by equation 2.11, to be respected.

Under these assumptions there may exist up to four different paths which all individually connect the two configurations. As illustrated in Fig. 3.9a, one possible path is given by following the right-handed arc until reaching the point where the tangent of the first circle c_{rs} coincides with the tangent of the second circle, which here is chosen to be the right-handed circle c_{re}. Following this straight line and then the right-handed arc on c_{re} then leads to passing through the chosen end point with the desired course. It can easily be seen that, given a sufficiently large radius R of the circles, this will be one possible path

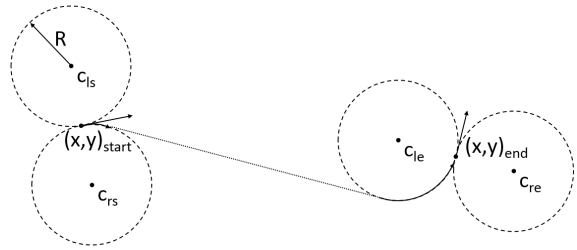
that does not violate the constraints of the system, yet achieves the desired configuration.

In the same way, the other paths, which can be seen in Fig. 3.9, are built up of a combination of choosing the right-handed and the left-handed circles. Although all of these paths are possible solutions to the problem, only one of them is defined as Dubins path – the one with the shortest travel distance. As such it is necessary to compute the lengths of each of the paths in order to determine which is Dubins path[2]. In this particular example it can be seen from Fig. 3.9 that Fig. 3.9b displays the shortest path.

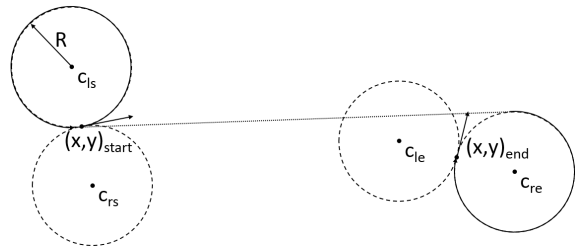
[2]Computing the lengths of the paths is beyond the scope of this thesis, but this is thoroughly explained by Beard and McLain [9].



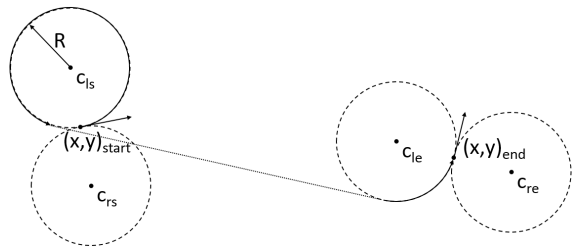
(a) Right-handed arc – Straight – Right-handed arc (RSR)



(b) Right-handed arc – Straight – Left-handed arc (RSL)



(c) Left-handed arc – Straight – Right-handed arc (LSR)



(d) Left-handed arc – Straight – Left-handed arc (LSL)

Figure 3.9.: Given a start and an end configuration, there exists four paths that each consists of an arc, a straight path and a second arc.

After determining which path is defined as Dubins path, the next step is to guide the UAV through this path. One way to do this is to compute the positions at which the vehicle exits and enters the circles. Such a method makes it possible to simply decide whether to follow a straight path or to follow a given circle with a constant radius.

One question that may arise is how to determine whether the vehicle has reached the designated point. Naturally, the ideas presented in sec. 3.2.5, also cover this matter. As mentioned, the halfplane method guarantees that the point has been reached. Hence crossing a computed halfplane will here signalize that the position has been reached. These halfplanes are illustrated in Fig. 3.10 at the points where the vehicle must change from circle to a straight line and vice versa.

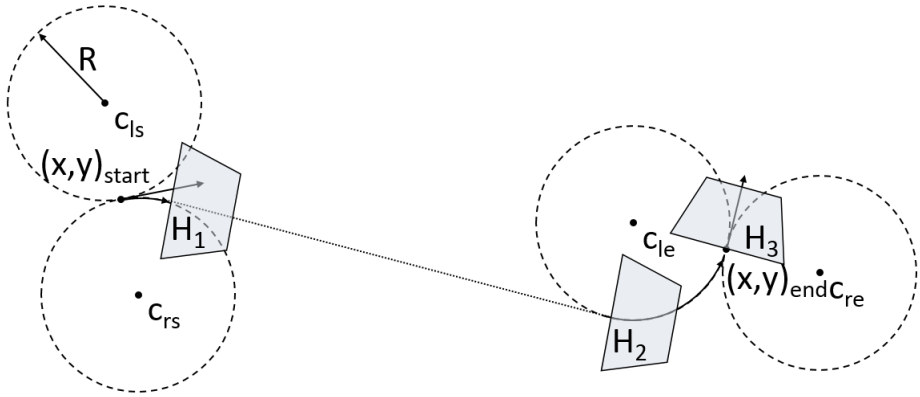
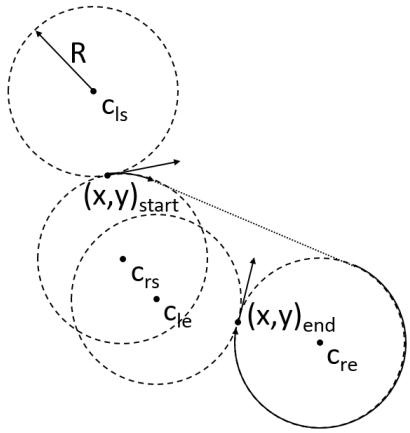


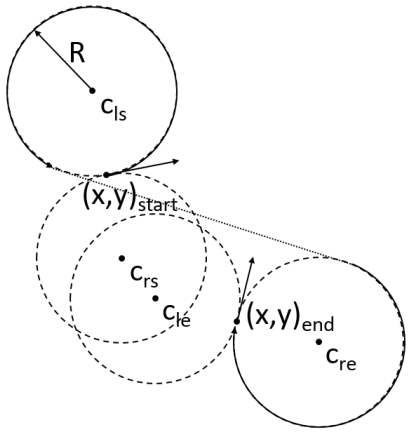
Figure 3.10.: Dubins path with halfplanes given as the positional success criterion.

Dubins path manages successfully to compute a possible transition from one configuration to another configuration without violating any constraints. However, one major drawback of the method, as presented by Beard and McLain [9], is that it requires the initial and terminal positions to be separated by at least a distance of $3R$ if all

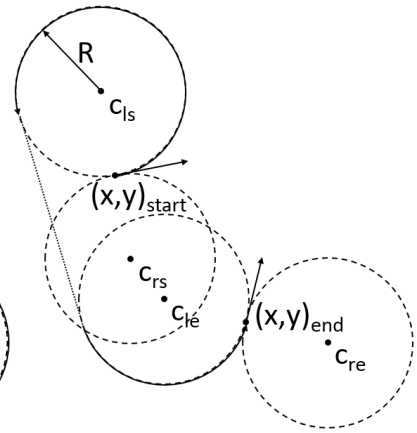
four paths are to exist. The reason for this limitation can easily be understood when considering the case displayed in Fig. 3.11. Here it is only possible to find three different paths. In fact, it is easy to observe that only two paths, namely RSR and LSL, can be guaranteed to exist.



(a) Right-handed arc – Straight – Right-handed arc (RSR)



(b) Left-handed arc – Straight – Right-handed arc (LSR)



(c) Left-handed arc – Straight – Left-handed arc (LSL)

Figure 3.11.: Displaying the possible paths when the distance between the two configurations is less than $3R$.

Moreover, there are two degenerate cases which will not be considered as they are very unlikely to happen. First, there is the case where the initial and terminal configurations lie on the same line and point in the same direction. This does not require a movement along an arc. Secondly, there is the case where both of the configurations lie on the same circle with the directions both going either clockwise or counter-clockwise. Here it is not necessary to move along a straight line.

3.3.3. Augmented Dubins Path

As presented, Dubins path, although providing an optimal transition between two configurations, possessed an important limitation in that it demanded a minimum distance between the two positions. This had to be considered if the method was to be used in this system. However, this limitation could be ignored by, instead of seeking four solutions, only look for the two solutions that always exist – RSR and LSL. Under this assumption it was possible to create a path which was based on the concept of Dubins path, namely the Augmented Dubins path (ADP).

3.3.3.1. Initial Idea

The idea behind this method was to assume an initial loitering around the desired point of impact $(x, y)_{target}$ with an arbitrary loitering radius R, equal to or exceeding the minimum turn radius. Upon initiation, the UAV would continue to follow the initial circle until reaching the straight line tangential to both of the circles, as seen in Fig. 3.12. It would then leave the circle $\mathbf{c}_1$ and move on to track the line until reaching the intersection between the line and the circle $\mathbf{c}_2$. At

this point the UAV would continue to track the arc until reaching the CARP $(x, y)_{release}$ where it would activate the release mechanism.

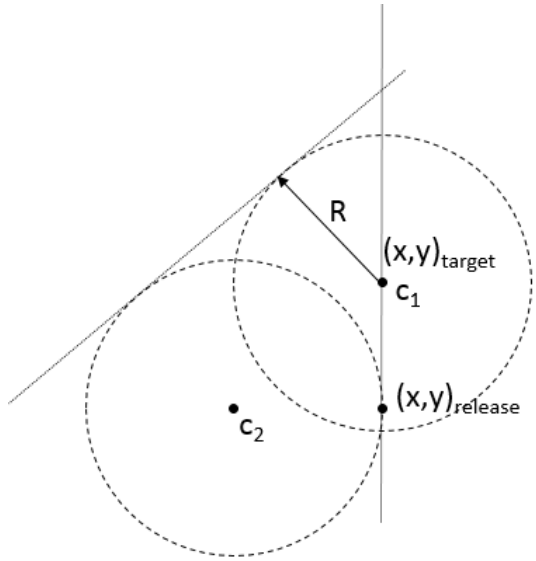


Figure 3.12.: Initial ADP concept.

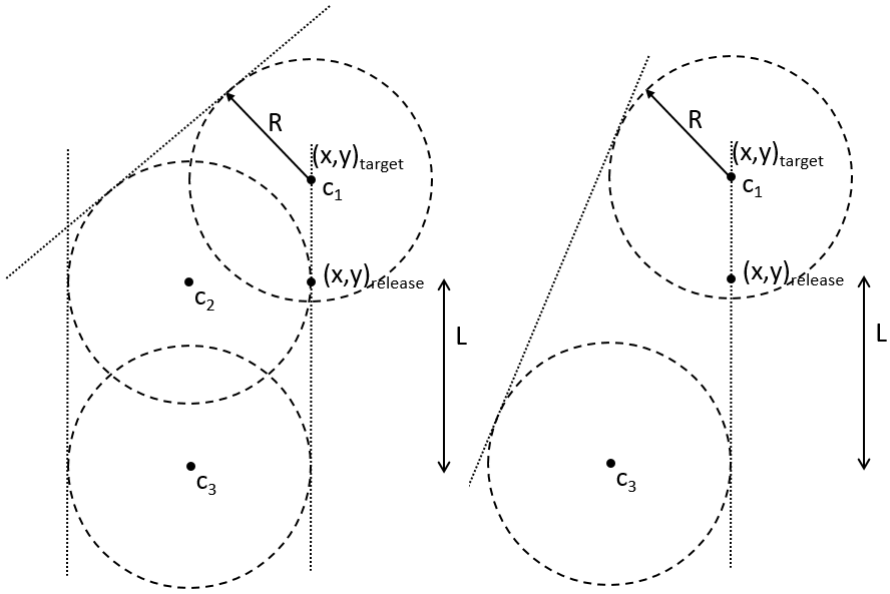
The initial idea did, however, possess some potential erroneous behavior because the CARP would lie on a circle. This meant that even though the UAV would have the correct course at the correct position, it would only stay in this configuration for a small time instant. In an ideal case, this would be sufficient. However, when considering the disturbances in the real world, such as wind, and possible deviations from the desired path, it would seem necessary to improve this path.

3.3.3.2. Improved Path

A possible solution to the aforementioned issue is to simply allow the UAV to travel along a straight path before approaching the CARP. This modification of the initial idea has been illustrated in Fig. 3.13a where it can be seen that a third circle $\mathbf{c}_3$ was added. This circle was placed a length L below $\mathbf{c}_2$, where the value of L depends on how much time the UAV should be allowed to straighten up before reaching the CARP. It can easily be seen, however, that upon applying this modification, the circle $\mathbf{c}_2$ becomes unnecessary. As such, a final concept, where the circle $\mathbf{c}_2$ is omitted, was developed. This method can be seen in Fig. 3.13b.

Another improvement that was considered was to compute the radius R in such a manner that it increased according to the magnitude of the wind, as shown in equation 2.12. This way, with an increasing wind, the bank angle would not saturate, which further would make the aircraft able to track the path more reliably. It was also decided to increase L in accordance with the wind. This was done by summing the IAS with the wind magnitude and further multiply this sum by a constant found through simulations to be sufficiently large.

In the final design the idea was to have the UAV create a path on the right-handed side or the left-handed side depending on whether the UAV was initially loitering clockwise or counter-clockwise around $\mathbf{c}_1$.



(a) Adding a third circle $\mathbf{c}_3$ at a distance L from $\mathbf{c}_2$.

(b) Removing $\mathbf{c}_2$ to get the final path.

Figure 3.13.: The ADP was improved by adding a straight line approach before reaching the CARP.

3.3.3.3. Concluding Remarks on the Augmented Dubins Path

As it was previously explained, this method bases itself upon the concept used in Dubins path in that it seeks to find a time-optimal path. However, in order to use this method it is necessary to generate more intermediate waypoints than the SLA. Nevertheless, in doing so, this method will, to a much larger degree, be able to control the UAV along the entire path. Thus it is evident that the ADP is likely to provide a far higher degree of determination as to what the resulting path becomes. Consequently, it was assumed that this method would outperform the SLA.

3.4. Path Tracking

Following the development of the desired paths, it was necessary to more closely examine the possible ways of controlling the UAV in order to track these paths. It was previously mentioned that the Piccolo SL can receive two different kinds of control inputs which can, individually, be used in order to control the position and the orientation of the Penguin B in the horizontal plane. In this section, these two control inputs, namely waypoints and heading, will be further examined and considered for usage in the final system.

3.4.1. Waypoints

By providing the Piccolo SL with waypoints it is possible for the Penguin B to track these waypoints in the order they were received. However, it is only possible to provide the autopilot with three waypoints at a time. Additionally, the final waypoint is set in the Piccolo DUNE task by default to be a safe loitering waypoint located a certain distance off, along the extension of the second waypoint. This further implies that any control strategy utilizing more than two waypoints, will need to provide the autopilot with new waypoints throughout the mission.

When dispatching waypoints on the IMC bus they will be interpreted in the Piccolo DUNE task which then sends the waypoints in the correct format to the Piccolo SL Autopilot. This process allows the autopilot to control the aircraft with an unknown algorithm. As such it may be difficult to determine exactly where the UAV intends to fly in order to reach the waypoints, as the exact path is not displayed in the PCC HMI. Moreover, it is not always possible to determine at what

angle it will approach the waypoint. Another issue that may arise, when utilizing waypoints as the control input, is that the Piccolo SL success criterion for reaching a waypoint may not necessarily satisfy the need for accuracy that this task desires.

3.4.2. Heading

Another way of controlling the aircraft is by continuously providing new heading references to the Piccolo SL Autopilot. By utilizing the kinematic model given by equation 2.6, with the input constraint on the heading turn rate as shown in equation 2.7, it is possible to design a controller that computes the desired heading.

A PID controller was considered, however, due to the simple nature of this controller, it was assumed that the Piccolo SL Autopilot waypoint control would outperform such a controller. Consequently, more advanced methods were pursued.

Ren and Beard [6] explained that the control design of nonlinear systems subject to input constraints can be approached by using either Model Predictive Control (MPC) or Control Lyapunov Functions (CLFs). Thus, these two methods will be briefly presented in relation to this project.

3.4.2.1. Model Predictive Control

The MPC is explained in the following way by Mayne et al. [20].

> "Model Predictive Control is a form of control in which the current control action is obtained by solving, at *each* sampling instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant."

Furthermore, to visualize this idea, Foss and Heirung [21] have formulated a simple MPC algorithm as shown in Algorithm 3.1.

Algorithm 3.1 State feedback MPC procedure

for $t = 0, 1, 2, ...$ **do**
    Get the current state $\mathbf{x}_t$.
    Solve a dynamic optimization problem on the prediction horizon from t to $t + N$ with $\mathbf{x}_t$ as the initial condition.
    Apply the first control move $\mathbf{u}_t$ from the solution above.
end for

Singh and Fuller [7] presented a solution to navigate a UAV through an urban terrain by the use of MPC. A nominal trajectory was set which consisted of the initial and the final position, in addition to an intermediate position. The problem here was to navigate through these points, while at the same time avoiding collision with the urban obstacles, i.e. the buildings. This problem was solved by creating and solving a number of piecewise-convex constrained MPC problems. Each problem considers the current position and generates a convex cone in which the vehicle will safely avoid collision with the obstacles.

In the same way, a similar strategy was proposed in order to track one of the given paths. According to Singh and Fuller [7], a linear quadratic tracker with penalties on the input and the error relative to the reference trajectory, can be formulated. Thus the following cost function to be minimized was suggested.

$$J_t = \sum_{k=t}^{t+N-1} (\mathbf{y}_{k+1} - \mathbf{r}_{k+1})^T \mathbf{Q} (\mathbf{y}_{k+1} - \mathbf{r}_{k+1}) + u_k^T R u_k \tag{3.18}$$

where $\mathbf{Q} \in \mathbb{R}^{2x2}$ is a symmetric positive semidefinite matrix, which decides the impact on each controllable variable, while R is a positive scalar which penalizes the input. The controllable variables are given as

$$\mathbf{y} = [x_1, x_2]^T$$

In equation 3.18 each time step is denoted by t while the length of the predicting time horizon is given by N. u is the heading turn rate and $\mathbf{y}$ is the position, as given by equation 2.6. Furthermore, as was done by Singh and Fuller [7], the reference trajectory $\mathbf{r} = [\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_M]$, $M \geq N$ can be created by interpolating between the waypoints.

Due to the nonlinear nature of the kinematic model in equation 2.6, this problem is, however, a nonlinear problem. Thus, Singh and Fuller [7] suggested that the vehicle model should be linearized about a nominal trajectory in order to transform this problem into a convex quadratic programming problem, which is far easier to solve.

3.4.2.2. Control Lyapunov Function

Ren and Beard [6] studied more closely the use of CLFs to track a trajectory, given constraints on velocity and heading. This is very similar to the aforementioned problem, except that the airspeed in this project was assumed constant. Thus, CLFs may also be considered for trajectory tracking of a given path in this project.

In general, the Lyapunov theory is of significant use in stability analysis. The theory can be explained by first assuming an ODE without any control inputs, and an equilibrium point $x_{equilibrium}$.

$$\dot{x} = f(x) \tag{3.19}$$

If a continuously differentiable Lyaponuv candidate function $V(x)$ exists, which satisfies $V(x) > 0$ and $V(x_{equilibrium}) = 0$, and the time derivative $\dot{V}(x)$ can be shown to be negative for all x except $x_{equilibrium}$, i.e. $\dot{V}(x) < 0, \forall x - x_{equilibrium}$, then $x_{equilibrium}$ is guaranteed to be asymptotically stable [22].

The CLF theory is based on the expansion of the Lyapunov theory to account for a given input u. This implies that the Lyapunov candidate function becomes $V(x, u)$, which is positive definite, and the control objective is to find a feasible feedback input u, such that $\dot{V}(x, u) < 0, \forall x$ [23]. Furthermore, if the reference path satisfies the constraints in the vehicle model in equation 2.6, as the path planning design intends to do, it is possible to track the path by the use of a CLF. In the same way as presented by Ren and Beard [6], the control objective becomes to find feasible control inputs for u^c at every time instant, such that $\mid x_1^{ref} - x_1 \mid + \mid x_2^{ref} - x_2 \mid + \mid \psi_2^{ref} - \psi_2 \mid \rightarrow 0$ as $t \rightarrow \infty$.

3.4.3. Concluding Remarks on Path Tracking

In order to control the Penguin B by utilizing heading as control input, it is imperative that the current heading of the aircraft has a high degree of accuracy. However, Skjong and Nundal [24] discovered that the heading, which could be extracted from the Piccolo SL, is based on the GPS measurements. Lack of GPS local precision led to the conclusion that the heading measurements likely did not provide a sufficiently high degree of accuracy. As such, it was decided not to pursue any of the heading reference path tracking methods in this project.

Consequently, waypoints, although presumably providing a less degree of accuracy and predictability than heading control, was chosen as the control input to the Piccolo SL.

4. Implementation

This chapter will present the implementation of the system, both on the hardware side and on the software side. Due to the distribution of the tasks, as presented in sec. 1.1.4, this section will focus to a larger degree on the software side of the implementation. For information about hardware, beyond what is presented here, the reader is referred to Siri Mathisen [25].

4.1. Payload

The successful deployment of the beacon from the aircraft required a custom designed payload which could perform the necessary tasks. Thus a payload, which could easily be mounted on board the plane, was assembled. This sections intends to introduce the payload which is comprised of the release mechanism as well as the necessary hardware to perform computations and activate the release mechanism.

4.1.1. Release Mechanism

As previously explained, the idea was to deploy the beacon from the aircraft in free fall. Consequently it was necessary to procure a release mechanism that could reliably release the beacon from the Penguin B.

Three different devices were more closely examined in order to determine which one would prove to be the most suitable for this mission.

The Servoless Payload Release System is a simple release mechanism that can be connected directly to a standard radio receiver [26]. In other words, it requires a Pulse-Width Modulated (PWM) signal in order to activate. Moreover, as can be seen in Fig. 4.1a, the device has conveniently placed holes such that secure installation on the Penguin B, and on the released beacon, is possible.

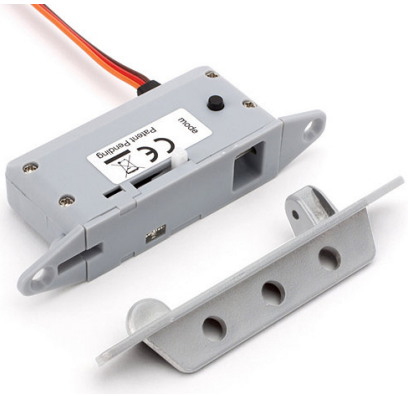
The Quanum RTR Bomb System possesses much of the same functionality as the Servoless Payload Release System, but as shown in Fig. 4.1b, this system is larger, and consequently also heavier. It can further be seen that accompanying this set is also a plate which can be mounted on a customized payload instead of using the included "bomb". However, the release mechanism does not possess any secure way of fastening it to the Penguin B.

The Peregrine Exhaustless CO2 Ejection System is designed such that the released payload would be ejected with an initial velocity caused by puncturing a CO2 cartridge [27]. This initial velocity creates a swift separation of the released payload and the aircraft. Moreover, the design of the device, illustrated in Fig. 4.1c, makes it ideal for installation inside the aircraft as it does not create a flame, nor does it leave any resulting residue.

4.1.1.1. Choice of Release Mechanism

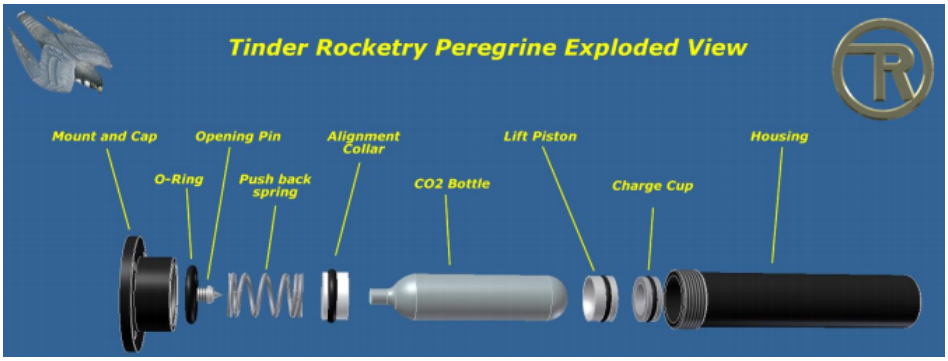
As explained in sec. 2.3.1, it was important to release the payload from the aircraft in such a way that it did not pose a threat to the aircraft after separation. For this reason the Peregrine Exhaustless CO2 Ejection System, offering an initial velocity, would be very desirable. However, it was decided that this device would possibly create more complications and was, as such, not chosen.

The remaining two options were both very similar. Still, the Servoless Payload Release System appeared more sturdy than the Quanum RTR Bomb System. Furthermore, it offered a practical way of securing the release mechanism to the aircraft. Consequently, the decision was made to use the Servoless Payload Release System for the initial test setup.



(a) The Servoless Payload Release System. [28]

(b) The Quanum RTR Bomb System. [29]



(c) The Peregrine Exhaustless CO2 Ejection System. [27]

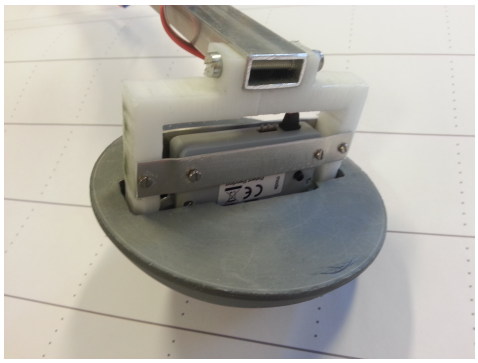
Figure 4.1.: The release mechanisms.

4.1.1.2. Mounting on the Penguin B

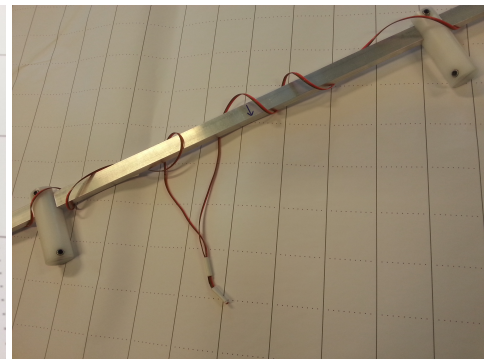
Upon having chosen a simple release mechanism, with no initial velocity, it was necessary to also mount the device safely on board the aircraft. It was imperative to mount it in a location which ensured that the deployed beacon would not collide with the propeller after separation from the aircraft. By mounting the release mechanism underneath the aircraft body, a safe separation could not be guaranteed. Consequently, the decision was made to mount two release mechanisms, for symmetrical reasons, on a bar which was then attached to the landing gear. Thus, actual in-air testing would imply dropping two identical objects simultaneously from the aircraft.

The mass of the release mechanisms, including the pair of drop-test objects, and the accompanying bar was negligible relative to the mass of the aircraft. Thus, it was assumed that the maneuverability of the aircraft was unaffected.

It can be seen in Fig. 4.2a how the release mechanism was secured to the bar, and further in Fig. 4.2b, how this bar was attached to the landing gear. The complete system can be seen in Fig. 4.2c. Moreover, when considering Fig. 4.2d, where the Penguin B is mounted on the launch catapult, it was possible to determine that the placement of the bar would not interfere with the launch from the catapult. It was, however, for landing purposes, desired to adjust the bar such that it was as close to the aircraft body as possible. In this way it would intervene as little as possible with the landing.



(a) Close-up of one of the installed release mechanisms (with gray, half-sphere drop-test object fitted).



(b) The white fasteners secured the bar to the landing gear on the Penguin B.

(c) Overview of the bar mounted on the Penguin B.

(d) The Penguin B, with the bar, placed on the launch catapult.

Figure 4.2.: Overview of how the release mechanisms were mounted on the Penguin B.

4.1.2. Penguin B Custom Payload

In sec. 2.3.1 the Penguin B was introduced and Fig. 2.4 reveals a so-called Universal Payload Mount installed underneath the aircraft. This mount can be equipped to fit the needs of any particular mission. As mentioned, there is, however, an upper weight limit to consider when assembling this payload. Moreover, it is also important to consider the limited space on the mount, as well as the placement of components, in order to avoid displacing the center of gravity (CG).

4.1.2.1. PandaBoard

The on board computer, responsible for performing the necessary computations, is a single-board computer known as a PandaBoard. The PandaBoard is booted from a Secure Digital (SD) card in a Linux-based operating system (OS). This system is intended to run the implemented DUNE software which utilizes the IMC messages to communicate both with the Piccolo SL and with the Ground Station.

4.1.2.2. PWM Circuit Board

As previously mentioned, the release mechanism requires a PWM signal in order to activate. The PandaBoard, however, does not provide an option to create this signal. As such it was necessary to create a circuit board that would create a PWM signal based on a high/low voltage input from a pin on the Pandaboard. This circuit board was set up to output a PWM signal, which activated the release mechanism, for as long as the pin was set to low. Upon changing the pin to high, the generated PWM signal would deactivate the release mechanism.

4.1.2.3. Communication

In order for all the devices in the system to function together it was necessary to set up a communication link between all the devices in the system. The devices are connected as illustrated in Fig. 4.3. It can here be seen that on board the Penguin B the Piccolo SL is connected to the PandaBoard via a serial cable, on the Piccolo SL end, and converted to a USB cable, on the PandaBoard end.

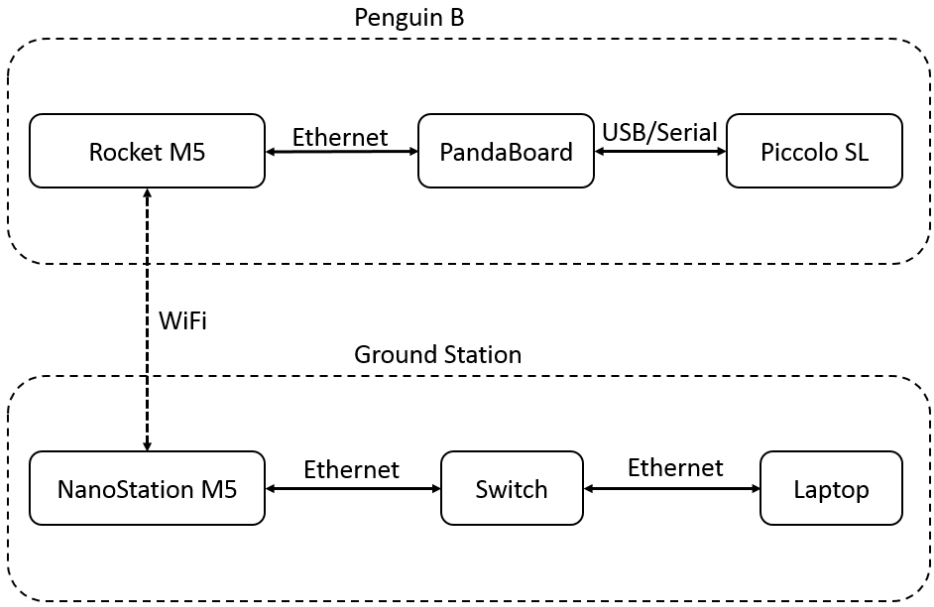


Figure 4.3.: Overview of the communication link between the devices in the system.

Furthermore, to achieve communication between the PandaBoard and the laptop, also known as the Ground Station, it was necessary to set up a wireless link. This communication link was set up by connecting a Rocket M5 to the PandaBoard and a NanoStation M5, via the Local Area Network (LAN), to the laptop.

Rocket M5 features a high-speed and long distance WiFi connection [30]. It does, however, require the usage of a pair of antennas in order to provide a sufficient range. The Rocket M5 can communicate both with another Rocket M5 and with a NanoStation M5.

NanoStation M5 acts as a client device and is set up to communicate with the Rocket M5 [31]. However, the NanoStation M5, unlike the Rocket M5, does not use external antennas and consequently it has a much shorter range. For this reason, during test flying at Agdenes, a Rocket M5 served as the communication unit on the ground as well.

4.1.2.4. Power Supply

In order to power the payload it was necessary to find appropriate power supplies for all of the components in the system. Initially it was assumed possible to power the entire payload with the serial cable from the Piccolo SL which provides a 12 V output. Thus a power converting circuit was designed and made in order to provide the components with the correct input voltage.

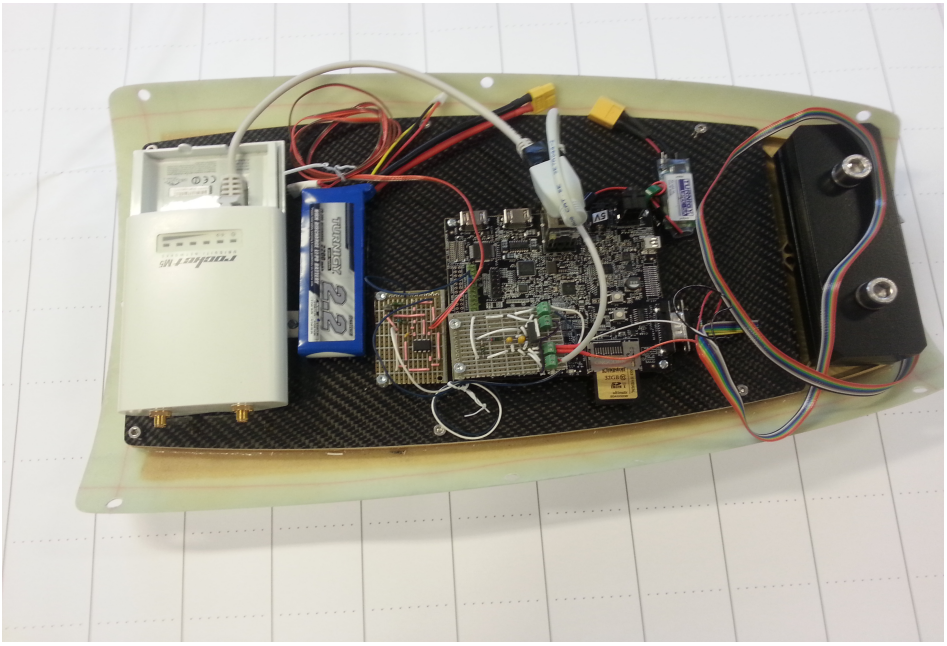
However, the PandaBoard requires a stable 5 V input, whereas the linearly down converted power from the serial cable was not sufficiently stable. Consequently, it was decided to power the PandaBoard by using a 7.4 V battery and further utilize a reliable voltage regulator to obtain a stable 5 V output. The Rocket M5 and the PWM circuit board, on the other hand, were powered by the aforementioned power circuit.

4.1.2.5. Hardware Architecture

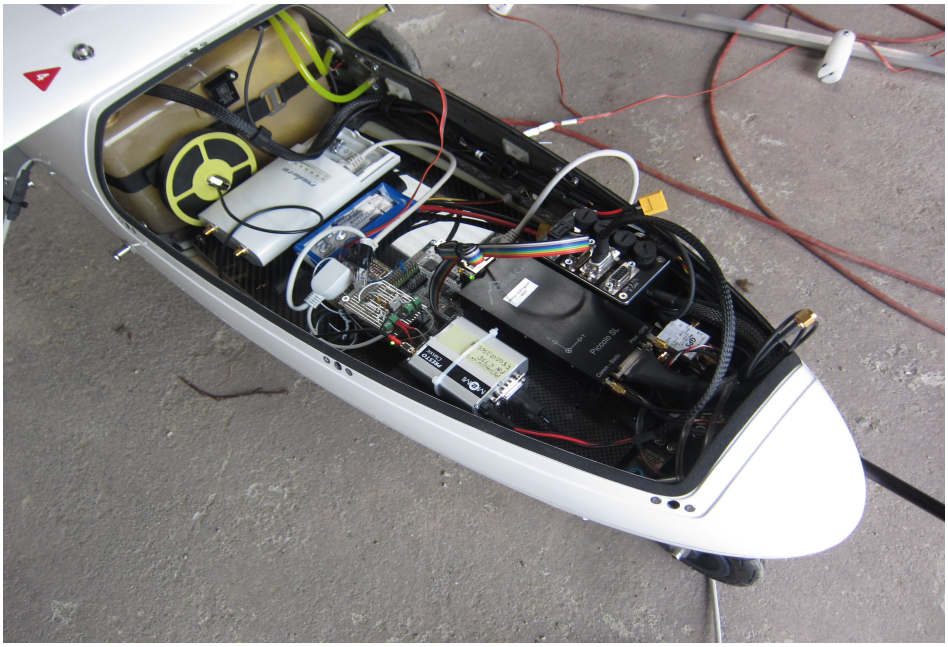
Lastly it was necessary to assemble the payload and install it on the Universal Payload Mount. The resulting hardware design would ideally be both well-presented and avoid any unnecessary tension on any of the cables. Furthermore, it was desirable to place the components in such a way that, once installed on the Penguin B, it would be easy to connect and disconnect any cables if deemed necessary.

Initial Setup The initial design was simple and sought to fulfill the aforementioned criteria, as can be seen in Fig. 4.4a, where the components have been installed on the Universal Payload Mount. However, unforeseen changes in the design of the Penguin B hardware resulted in the need to alter the initial hardware setup. Still, Fig. 4.4b displays how the payload appeared when mounted on the Penguin B.

Improved Setup In the improved hardware setup it was decided to utilize a metal casing fastened to the Universal Payload Mount. The components would then be installed on this metal casing. This would not only ensure that the payload could fit in the Penguin B, but also provide a better framework for future work on the payload. It can be seen in Fig. 4.5 how the Rocket M5 was installed on top of the lid for an easy way to connect it to the antennas. The PandaBoard was placed underneath the lid to avoid any tension on the ethernet cable to the Rocket M5. Further, the PWM circuit board was installed on the side of the metal casing. This left room for the battery and the voltage regulator to be attached with Velcro to the floor of the Universal Payload Mount.

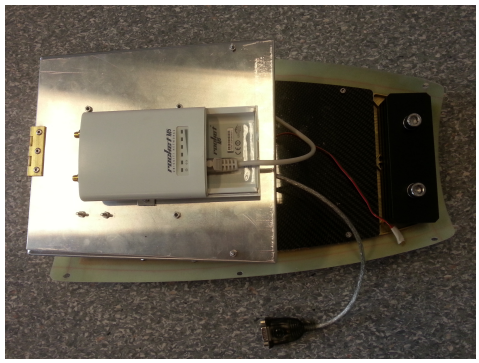


(a) Setup of the hardware.

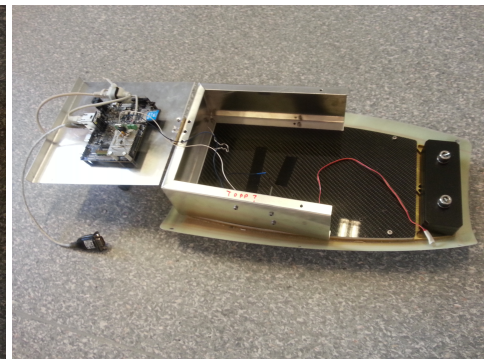


(b) Setup of the hardware when mounted on the Penguin B.

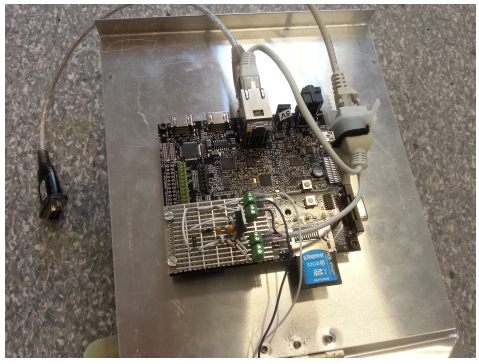
Figure 4.4.: Overview of the initial setup of the hardware.



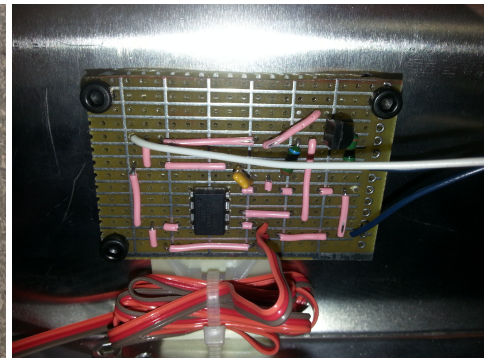
(a) The Rocket M5 installed on top of the casing.



(b) Overview of the open casing.



(c) The PandaBoard installed underneath the lid.



(d) The PWM circuit board installed on the side of the casing.

Figure 4.5.: Overview of the improved setup of the hardware.

4.2. Software Implementation

This chapter will more closely examine the necessary software which was implemented in order to achieve the desired functionality with the provided framework. As previously mentioned, this software was implemented in C++ and modulated into classes and DUNE tasks. A schematic overview of the system can be seen in Fig. 4.6. In this figure the solid arrows indicate communication via the IMC messages, while the dotted lines indicate a call to another class. Throughout the presentation of the implemented software, this figure is intended to provide a better understanding of how the system, as a whole, is composed. A more detailed description of the specific classes may be found in Appendix B.

4.2.1. GroundUnit

The GroundUnit has one simple purpose, namely initiating the precision airdrop system. In this task the desired point of impact is set in the WGS84 coordinate system. Furthermore, the IMC::Target message is generated to contain the given target coordinate and a specific destination label such that only the chosen control algorithm is initiated. This label is set to either initiate the SLA or the ADP.

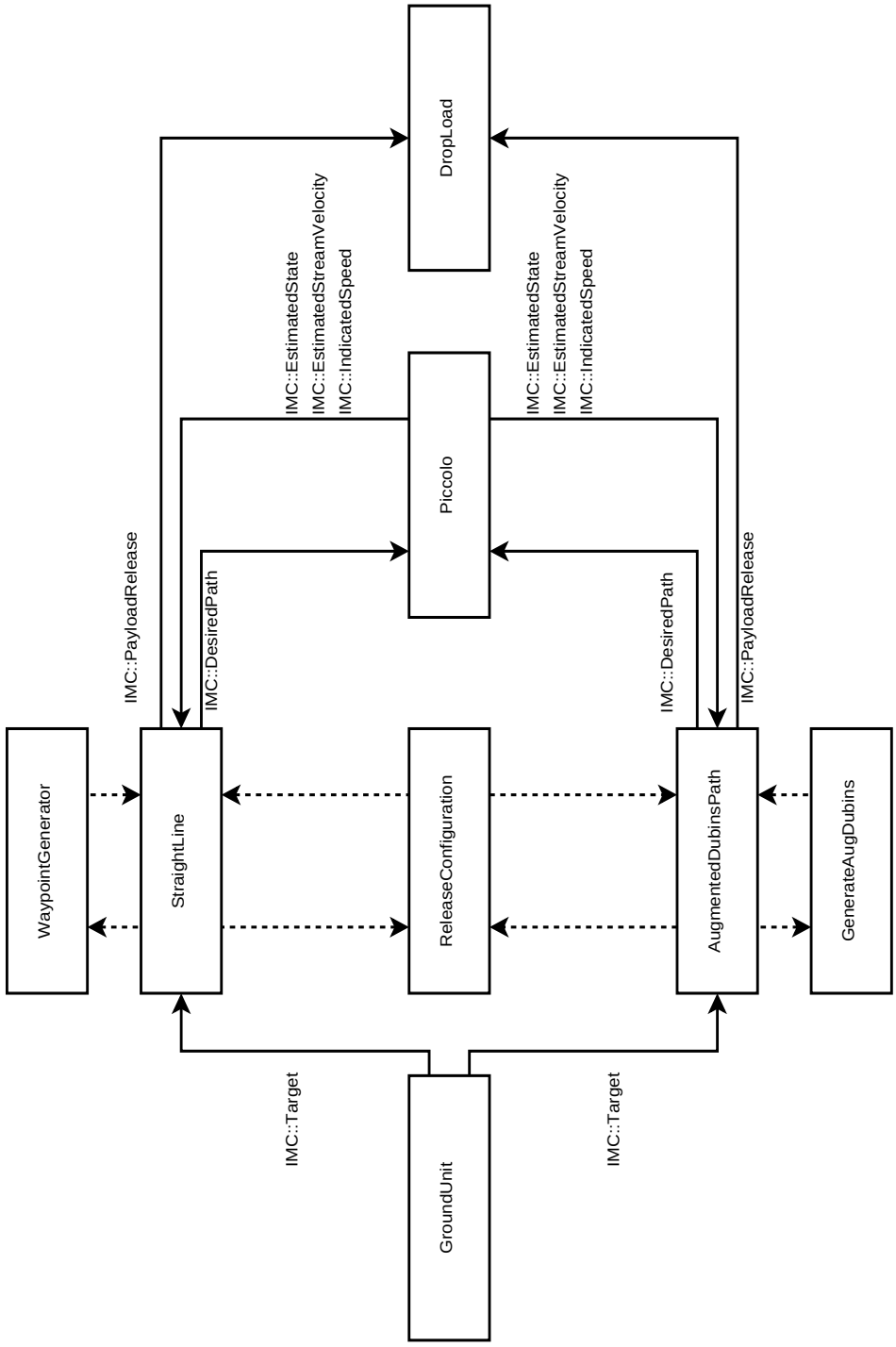


Figure 4.6.: Overview of the relationship between the DUNE tasks and the classes.

4.2.2. CARP

This section will introduce how the computation of the CARP was modulated into two different classes, namely ReleaseVelocity and ReleasePoint, and further how these classes were combined to obtain the class ReleaseConfiguration, as seen in Fig. 4.7.

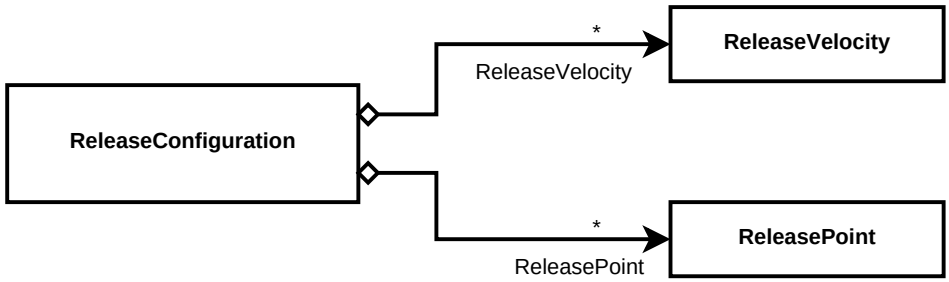


Figure 4.7.: The ReleaseConfiguration is comprised of the classes ReleaseVelocity and ReleasePoint.

4.2.2.1. ReleaseVelocity

It was previously explained that the beacon should be released in the direction opposite to that of the wind. This code will assume that the wind vector has already been found. Moreover, in accordance with the assumption made in Equation 2.13, it will also be assumed that the IAS is known. Following these assumptions, the ReleaseVelocity class will calculate the inverse wind unit vector, i.e. the unit vector in the desired direction of flight, and further use this to calculate the release velocity vector.

4.2.2.2. ReleasePoint

This class intends to solve the system of ODEs, given by equation 3.11, before applying the translational shifting of the points as explained in

equations 3.12 through 3.15. The ODE system was solved by utilizing functionality in the Boost C++ Libraries along with a defined initial state vector. This vector is, as shown in equation 3.11, a combination of the release velocity, acting as the initial velocity of the free fall, and the target horizontal position, acting as the tentative release position, along with the release altitude. Additionally, the wind vector was included in the calculations in order to account for the effects of the wind during the free fall.

4.2.2.3. ReleaseConfiguration

The purpose of the ReleaseConfiguration class was primarily to combine the two aforementioned classes into one class. Additionally, however, it also sought to solve the issue wherein the wind was absent or negligible. As the direction based itself on the direction of the wind, this would pose a problem if the wind was not present. Consequently, this class has a function which solves this issue by choosing the default wind direction to be from south to north. Thus the default direction of approach would be from the north.

A call to the ReleaseVelocity class is made to obtain the release velocity, which along with the target position and release altitude, comprise the initial state vector. This vector is then used in the ReleasePoint constructor, which solves for the CARP. The final output of the ReleaseConfiguration is a vector consisting of the wind unit vector and the CARP. Although the release configuration ideally should consist of the release position and the release direction, it was deemed more useful to output the wind unit vector instead.

4.2.3. Control Algorithms

As Fig. 4.6 illustrates, there are two different control algorithms which may be used to guide the aircraft into the desired configuration. Both of these options are modulated into one DUNE task and one associated class.

4.2.3.1. Straight Line Approach

The SLA was implemented as an initial and simple solution to the primary objective of the project, namely achieving the correct configuration and releasing the beacon at the CARP. It consists of the WaypointGenerator class and the StraightLine task.

WaypointGenerator The WaypointGenerator class is responsible for producing two waypoints that are aligned in the desired direction of flight. This is accomplished by creating a point which is shifted a certain distance in the direction of the wind unit vector. The shifting constant, which along with the minimum turn radius, determines this distance, was set according to sec. 3.3.1.1.

StraightLine The StraightLine task is initiated by consuming the IMC::Target message with the SLA destination label. A call to the ReleaseConfiguration class is made to obtain the CARP, which is used in the WaypointGenerator to obtain the two waypoints. These waypoints are then dispatched on the IMC::DesiredPath message, which is picked up in the Piccolo task and forwarded to the Piccolo SL Autopilot.

Following the initiation of the path, a function in this task is designed to continuously check the set release criterion. Upon validating the

release criterion, an IMC::PayloadRelease message is sent which activates the release mechanism.

4.2.3.2. Augmented Dubins Path

The ADP was the second control algorithm which was implemented, and was designed to be the most optimal control that could be achieved with waypoints as the control input. This control is comprised of the GenerateAugDubins class and the AugmentedDubinsPath task.

GenerateAugDubins This class is responsible for creating the desired path in accordance with the previously introduced theory on the ADP. The circle centers $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$, as seen in 3.13a, are found according to Algorithm 4.1. Here the radius R of the circles is determined according to equation 2.12. Furthermore, the rotation matrix $\mathcal{R}_z$ is set to rotate $\pm\frac{\pi}{2}$, depending on whether the method is set to move counter-clockwise or clockwise.

The halfplanes, which constitute the desired path, are then found by using Algorithm 4.2 on the circle pairs $(\mathbf{c}_1, \mathbf{c}_3)$ and $(\mathbf{c}_3, \mathbf{c}_2)$. A list of four elements, wherein each element contains a halfplane and the associated circle center, is then returned to the executing task.

Algorithm 4.1 Calculating the circle centers

Input: Target position $\mathbf{x}_{target}$, Release position $\mathbf{x}_{release}$, Wind unit vector $\mathbf{e}_w$, Length L

$\mathbf{c}_1 \leftarrow \mathbf{x}_{target}$

$\mathbf{c}_2 \leftarrow \mathbf{x}_{release} + R\mathcal{R}_z\mathbf{e}_w$

$\mathbf{c}_3 \leftarrow \mathbf{c}_2 + L\mathbf{e}_w$

Output: $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$

Algorithm 4.2 Computing the halfplanes

Input: Circle centers $\mathbf{c}_{start}$, $\mathbf{c}_{end}$

$\mathbf{q}_1 \leftarrow \frac{\mathbf{c}_{end}-\mathbf{c}_{start}}{\|\mathbf{c}_{end}-\mathbf{c}_{start}\|}$

$\mathbf{q}_2 \leftarrow \mathbf{q}_1$

$\mathbf{z}_1 \leftarrow \mathbf{c}_{start} + R\mathcal{R}_z\mathbf{q}_1$

$\mathbf{z}_2 \leftarrow \mathbf{c}_{end} + R\mathcal{R}_z\mathbf{q}_1$

$\mathcal{H}_{start} \leftarrow (\mathbf{z}_1, \mathbf{q}_1, \mathbf{c}_{start})$

$\mathcal{H}_{end} \leftarrow (\mathbf{z}_2, \mathbf{q}_2, \mathbf{c}_{end})$

Output: $\mathcal{H}_{start}$, $\mathcal{H}_{end}$

AugmentedDubinsPath The ADP is, naturally, different from the SLA, although some of the functionality remains the same. This task, too, is initiated by the GroundUnit task dispatching an IMC::Target message with the appropriate destination label. However, upon receiving this message, the ADP initiates the state machine which can be seen in Fig. 4.8. An explanation of each state is described below.

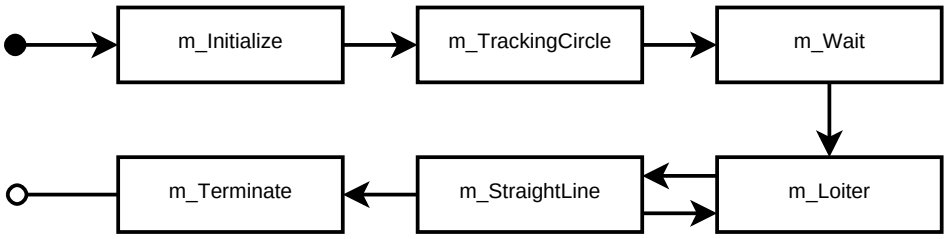


Figure 4.8.: State machine in ADP.

m_Initialize initializes the control of the Penguin B by calling the ReleaseConfiguration class to obtain the CARP. Next, it receives a list, containing the desired path, from a call to the GenerateAugDubins class. Lastly it sets the vehicle to initialize loitering around the set target with a radius given by GenerateAugDubins.

m_TrackingCircle continuously checks the distance from the current position to the previously defined loitering circle. As an extra insurance, this state also checks that the vehicle loiters in the correct direction. Upon fulfilling these requirements, the state changes to m_Wait.

m_Wait was included because the ADP uses the halfplane criterion to check the position relative to the desired position. To avoid undesirable behavior, the vehicle stays in this state until it loiters at the correct side of the circle, upon which the state changes to m_Loiter.

m_Loiter ensures that the vehicle tracks the current circle arc until the designated halfplane has been reached. At this point it extracts the next element from the list and enters the straight line segment, consisting of two waypoints, as seen in Fig. 3.13b, before changing to the state m_StraightLine.

m_StraightLine tracks the current straight line segment until the designated halfplane has been crossed. Next it generates and tracks a circle, based on the next extracted list element, and changes state to m_Loiter. If, however, there is only one element left in the list of halfplanes, the state is changed to m_Terminate.

m_Terminate simply indicates that the vehicle has entered the last straight segment.

After the state machine has been terminated, the task continuously checks to see if the set release criterion has been fulfilled. Upon validating this criterion, the IMC::PayloadRelease message is sent and the release mechanism is activated.

Adjusting halfplanes Testing revealed that the Piccolo SL was always given three waypoints – an initial waypoint, a final waypoint and a third default waypoint in the extension of these. Initially, when crossing a halfplane, the idea was to utilize the halfplane point **z** as the initial waypoint. Further, depending on the state, the second waypoint would either be set to a loitering circle or a waypoint creating a straight line segment. However, due to some unexpected Piccolo SL behavior, it was not possible to set the initial waypoint in this manner. Thus, the decision was made to move the halfplanes tangentially to a point before the initial point.

4.2.4. Payload Release

In order to activate the release mechanism, the release criterion has to be fulfilled. This section introduces the task related to activating this mechanism as well as a task which was used to test the different release criteria.

4.2.4.1. DropLoad

The DropLoad task receives an IMC::PayloadRelease message from either the SLA task or the ADP task, depending on the chosen path. The contents of this message could either instruct the DropLoad task to activate or deactivate the release mechanism. Upon receiving the instruction to release, this task would set the voltage of a designated pin to low, and consequently the release mechanism would be activated, in accordance with sec. 4.1.2.2. In a similar way, when the message instructed the deactivation of the release mechanism, this task would set the pin to high and the release mechanism was deactivated.

4.2.4.2. ReleaseObject

Although not explicitly a part of the final system, the release criteria functions were tested and implemented in a task known as ReleaseObject. The two criteria are, as previously mentioned, the circle criterion and the halfplane criterion. This task would simply consume an IMC::Target message from the GroundUnit. The distance from the current position to this target position would then be measured, and the release mechanism would be activated if this distance was sufficiently small. In the same way, a halfplane criterion was implemented according to equation 3.17.

5. System Testing

In order to determine how well the system, as a whole, performed, it was necessary to conduct some experimental tests. This testing was carried out by simulations, both in the HIL simulator and in Matlab, in addition to field testing on the Penguin B at Agdenes Airstrip. In this chapter the conducted tests will be presented, both in terms of purpose and method, along with the results and an accompanying analysis.

5.1. Simulation

Before the Penguin B, and the implemented payload, could be tested on Agdenes, it was important to simulate the system in a safe environment. Through utilization of the HIL simulator it was possible to obtain a better understanding as to how the system would perform. This was, naturally, an incredibly useful tool during the implementation of the software. Upon successfully achieving a desired result, this data was also analyzed and used to determine the reliability of the system.

This section intends to first look at a simulation of the two different release criteria. Next, the performance of the two different control algorithms is more closely examined. And lastly, the expected ac-

curacy and precision of the free fall is presented. Throughout these tests, the IAS of the Penguin B was set to the desired 28 m/s, and the altitude was assumed to be constant during flight. Moreover, unless otherwise specified, the tests were performed without any form of wind disturbances.

5.1.1. Release Criteria

As previously mentioned in sec. 3.2.5, there were two different release criteria which were considered. Moreover, some advantages and disadvantages regarding both methods were presented. In this subsection, a simulation of the two methods will be presented in an attempt to determine their overall performance.

This simulation was set up in the HIL simulator by creating a flight path, consisting of multiple waypoints, where three of the waypoints were aligned in such a way that the aircraft would consistently approach the CARP as was illustrated in Fig. 3.6. The distance from the Penguin B to the CARP was logged as long as the vehicle was within the circle and as soon as the vehicle crossed the halfplane. This way both of the tests could be conducted simultaneously.

5.1.1.1. Circle

Testing revealed that the simulation sampled the position at a fairly high rate, and consequently, in order to avoid unnecessary large amounts of data, it was decided to set the radius of the circle to 10 m. As such, it can be seen in Fig. 5.1, that the aircraft entered the circle along the x-axis at -10 m, continuously logged the distance, and exited the circle at +10 m. A closer examination of Fig. 5.1 reveals that within each

circle there are approximately 20 logged samples. This procedure was repeated several times to ensure that this result was consistent.

Furthermore, the total horizontal distance was of significant importance in order to determine the smallest and largest achievable distance from the aircraft to the CARP. As revealed in Fig. 5.2, the minimum logged, total horizontal distance was emphasized, and it can be seen to consistently vary between 0.9 - 1.5 m. This result can be explained by looking at two different inaccuracies in the system.

First, it can be seen that the vehicle did not fly completely straight along the x-axis, which caused a deviation in the y-axis. This further resulted in a larger total horizontal distance. Secondly, the frequency $f_{logging}$ at which the position was logged limits the achievable precision. The maximum achievable deviation in the x-axis can be approximated by

$$x^e_{max} \approx \frac{\text{IAS}}{f_{logging}} \tag{5.1}$$

It can be seen in equation 5.1 that $x^e_{max} \to 0$ when $f_{logging} \to \infty$. Consequently, with a higher logging frequency, it is possible to achieve a higher accuracy in the measurements. In the same way, equation 5.1 also reveals that the accuracy is increased with a lower IAS.

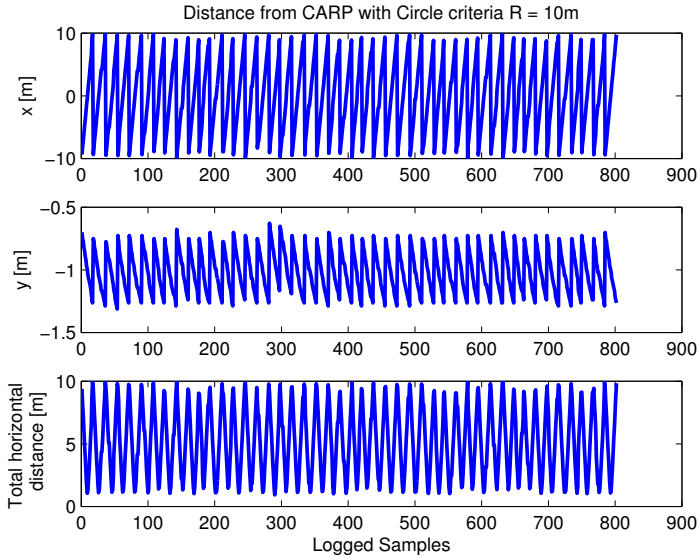


Figure 5.1.: The measured distance from aircraft to the CARP with the circle criterion.

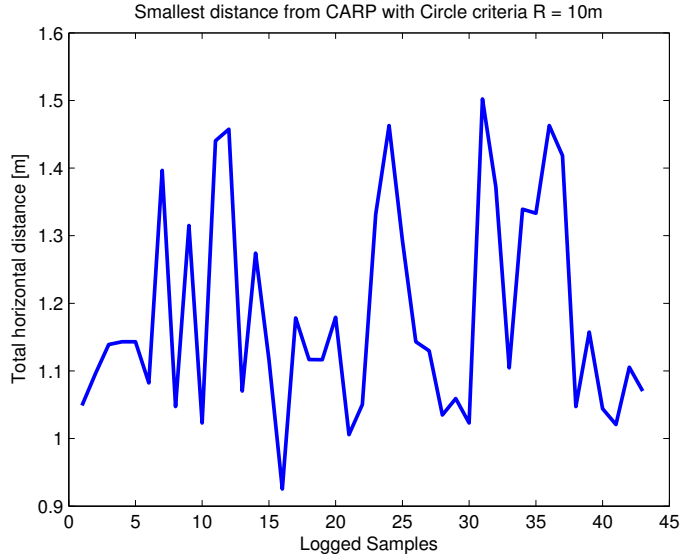


Figure 5.2.: The smallest measured total horizontal distance from the aircraft to the CARP with the circle criterion.

5.1.1.2. Halfplane

Unlike the circle criterion, where the closest logged point can appear both before and after passing the CARP, it is easily understandable that, with the halfplane criterion, the closest point can only appear after passing the CARP. It can be seen in Fig. 5.3 that the total horizontal distance ranges from 1.2 - 1.8 m. Here it can readily be observed that the error in the x-axis is, once again, caused by the limitation in logging frequency, while the error in the y-axis is caused by the same deviation from the flight path as previously explained.

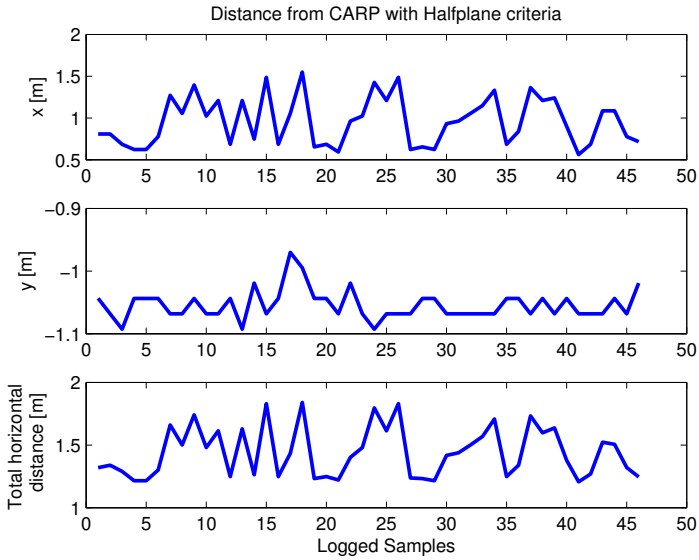


Figure 5.3.: The measured distance from the aircraft to the CARP with the halfplane criterion.

5.1.2. Path Simulation

As will later be shown in sec. 5.1.3, the performance of the system as a whole, depends largely on the performance of the control strategy used

to achieve the desired configuration. Thus, the decision was made to investigate how well the two implemented control strategies, the Straight Line Approach and the Augmented Dubins Path, performed in the HIL simulator.

These tests desired to examine the performance given different initial configurations. Additionally, by applying an artificial constant wind in the HIL simulator, it was possible to observe how this disturbance affected the overall tracking of the path and, more importantly, the configuration at the CARP. As mentioned before, in the absence of wind, the vehicle would approach the CARP from north to south. As such, in order to obtain a result that was easily interpretable, it was decided to let the simulated wind move from south to north, such that the aircraft approach would remain the same. The tests looked at three different scenarios – no wind, 5 m/s wind and 10 m/s wind.

Furthermore, a second set of tests were conducted in order to observe the consistency of the release configuration with the different control strategies. These tests were completed over a long time by allowing the methods to repeat themselves, with approximately the same initial configuration, after successfully crossing the halfplane defined by the CARP. Unlike the previous tests, these tests were conducted without any wind disturbances.

5.1.2.1. Straight Line Approach

The SLA creates two waypoints that are aligned and distanced in such a way that by letting the Penguin B track these, the vehicle would ideally approach the CARP at the correct angle. The HIL simulation of this strategy returned a valuable insight into the performance of the SLA. Moreover the distance d, as mentioned in sec. 3.3.1.1, was

set equal to four times the value of the minimum turn radius, as defined by equation 2.12.

It can be seen in Fig. 5.4 that regardless of what initial configuration the vehicle had, it moved towards the first waypoint. This waypoint, although not marked, can be seen as the upper point where all the paths coincide. Each of the paths approached this point at different angles depending on their initial configuration. This further resulted in different trajectories when transitioning from the first waypoint and towards the second waypoint, namely the CARP. Although all of the paths attempt to straighten up prior to arriving at this point, it is easy to observe that this is difficult to accomplish with the SLA. It is also worth mentioning that after reaching the second waypoint, i.e. the CARP, the aircraft moved towards the third automatically generated waypoint, a behavior which was expected as explained in sec. 3.4.1.

Furthermore, the results from the first wind test with the SLA can be seen in Fig. 5.5. This plot reveals that when the wind is applied, the vehicle receives a push when flying north, which consequently makes the turn at the first waypoint slightly more difficult. However, upon changing the heading towards south the aircraft is flying against the wind. Consequently the speed is decreased and the aircraft is able to maneuver more easily. Nevertheless, by increasing the wind further, Fig. 5.6 reveals that the vehicle receives a significant push when flying towards the first waypoint. This further makes it incapable, in some cases, of straightening up properly before the reaching the CARP.

Additionally, it can be seen from these figures that the end configuration may depend largely on the initial configuration. This is especially relevant for the cases with a wind disturbance as the initial configurations located south of the CARP achieves a far better end configuration than the initial configurations located north of the CARP. This

can be explained by understanding that the push of the wind effectively increases the distance d when flying north. Consequently the paths starting south of the CARP will have more time to straighten up before reaching the CARP. This result further supports the theory that the performance achieved with this method depends, to a large degree, on the distance d.

Moreover, the choice of approaching the CARP against the wind is also further supported when considering that the wind slows the aircraft down, which increases the maneuvering window. However, it is important to emphasize that this result does *not* imply that it is better to release the beacon in windy conditions, a point which will be further examined in sec. 5.1.3.

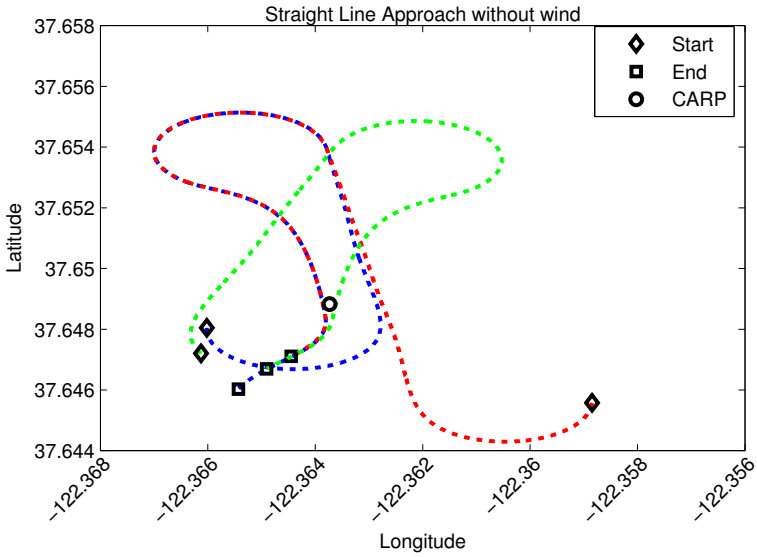


Figure 5.4.: UAV path tracking of the SLA without any wind.

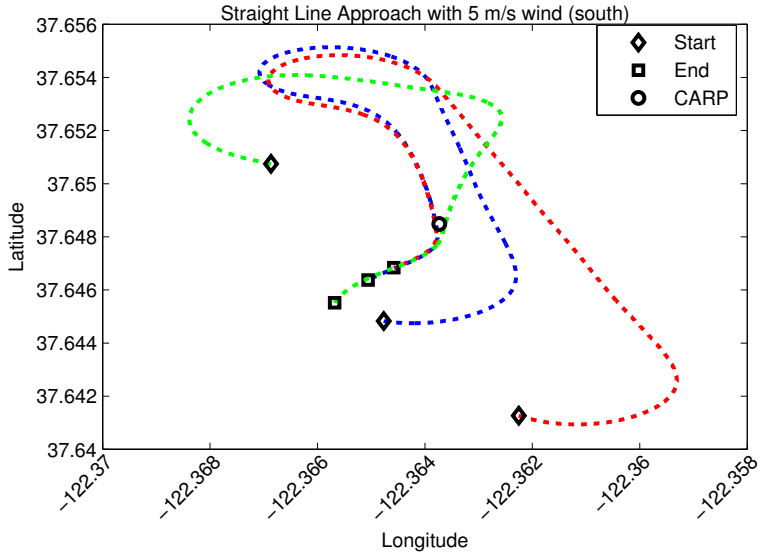


Figure 5.5.: UAV path tracking of the SLA with 5 m/s wind from south.

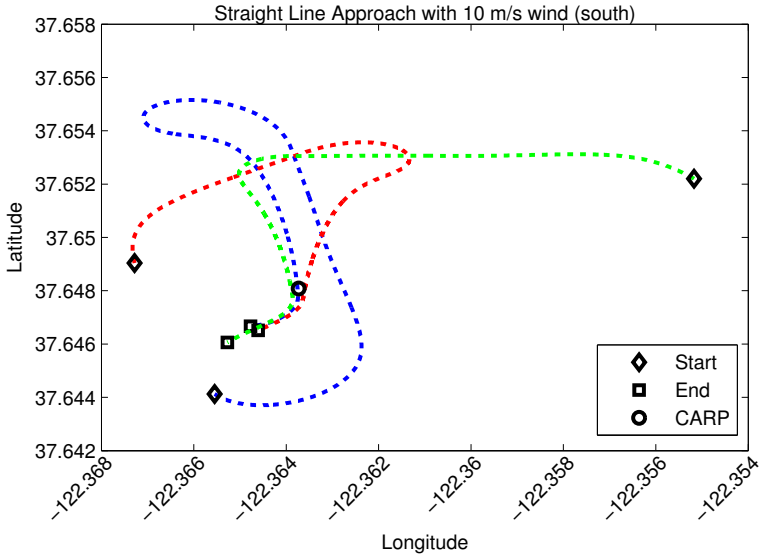


Figure 5.6.: UAV path tracking of the SLA with 10 m/s wind from south.

While the previous plots provided an understanding of how the aircraft moved, they did not give a proper insight into the actual accuracy of the end configuration. However, the following test ran the SLA several times and gathered the end configuration data which revealed how well the algorithm performed. From Fig. 5.7, it can be seen that the aircraft deviates in both the x- and y-axes from the desired position. As was previously presented, the deviation in x is primarily caused by the limitation in positional logging frequency.

The error in y is, however, caused by the aircraft being unable to accurately track the desired path and as such pass through the CARP. This behavior could be caused by the distance d being too small. On the other hand, another explanation suggests that the Piccolo SL Autopilot accepts a certain deviation from the waypoint and consequently does not attempt to accurately track the point.

Fig. 5.8 is an estimated normal distribution based on the deviation from the desired angle at the CARP. It can easily be seen that the aircraft approaches the CARP at an inaccurate angle. Similar to what was observed in the previous tests, the aircraft is clearly unable to approach the CARP with a higher angular accuracy.

Additionally, in both of these plots it appears that the end configuration is biased towards one direction. This bias is explained by understanding that the Piccolo SL desires to find the shortest path from the current position and to the next waypoint, without any consideration for the angle at which it approaches the waypoint. This behavior causes the aircraft to follow the same path throughout the entire simulation, and thus the error is biased to one side.

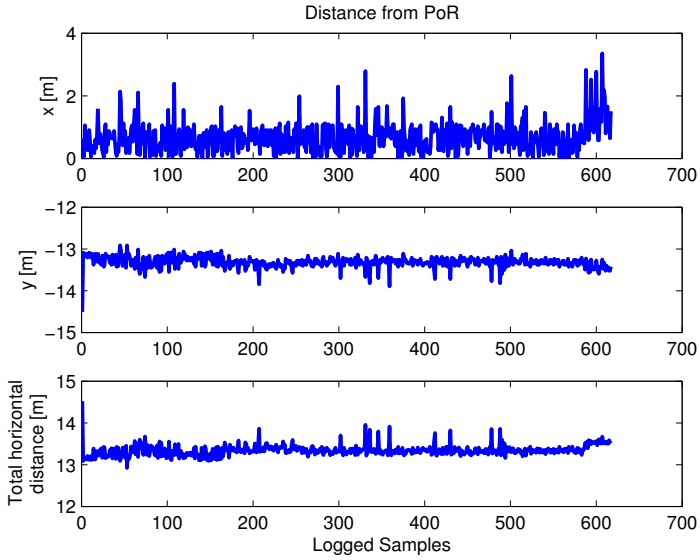


Figure 5.7.: Long time testing of the deviation from the CARP with the SLA.

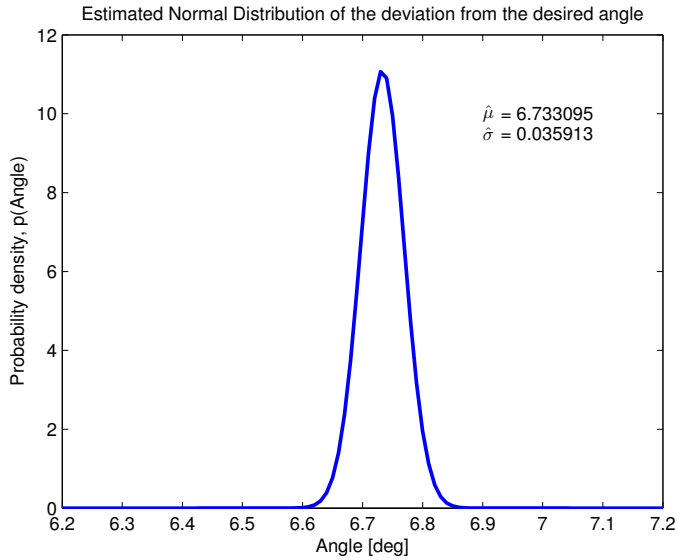


Figure 5.8.: Estimated Normal distribution of the deviation from the desired angle based on the measurements from the SLA.

5.1.2.2. Augmented Dubins Path

Because the ADP based itself on Dubins Path, which is the time-optimal transition between two configurations, it was assumed that this method would most likely out-perform the SLA. The length L, as introduced in sec. 3.3.3.2, was set to equal to five times the sum of the IAS and the magnitude of the wind. This control strategy was then tested in the HIL simulator.

Fig. 5.9 reveals how the aircraft, given different initial configurations, approached and commenced tracking of the first circle. Next, the aircraft left the circle at the desired point and followed a straight line until it was sufficiently close to the second circle. It can readily be seen that the aircraft then tracked this circle before exiting at a point aligned with the CARP. This way the aircraft was able to approach the CARP at a near optimal angle. After exiting the circle, the aircraft attempted to approach the line created by two waypoints. This way it was possible to straighten up the vehicle, if necessary, before reaching the CARP.

When wind was applied to the HIL simulator, it can be seen in Fig. 5.10, that the path did not change noticeably. This behavior is explained by understanding that the algorithm, to some degree, adapts to the wind magnitude. With an increase in the wind magnitude, the minimum turn radius was increased, and consequently the radii of the circles were increased. Furthermore, as previously mentioned, the size of L was also increased in accordance with the wind magnitude. This allowed the vehicle a longer time to straighten up after exiting the final circle.

By increasing the wind further, as shown in Fig. 5.11, it was shown that the algorithm struggled to track the expected path in the different

instances. However, the general shape of the path can still be seen and upon approaching the CARP, all of the paths coincided and were able to achieve a satisfactory final configuration.

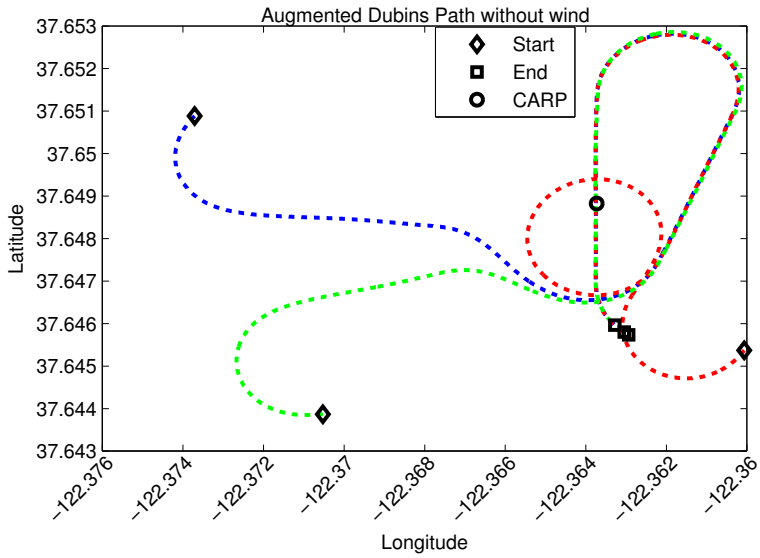


Figure 5.9.: UAV path tracking of the ADP without any wind.

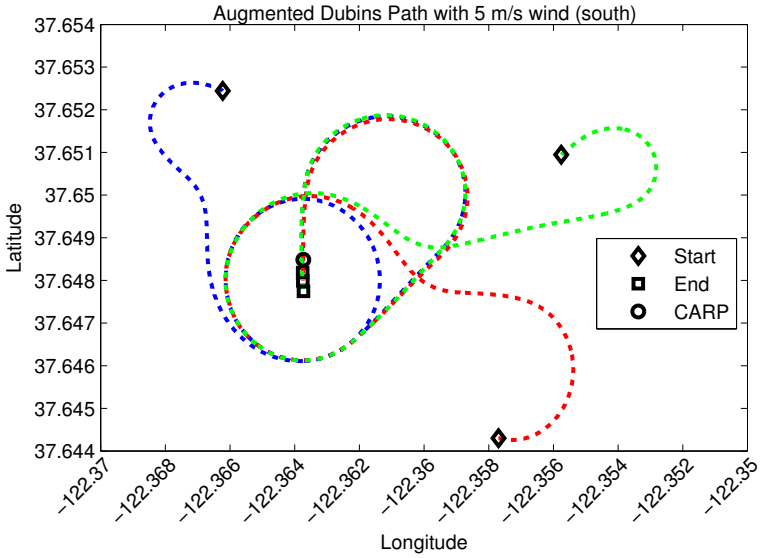


Figure 5.10.: UAV path tracking of the ADP with 5 m/s wind from south.

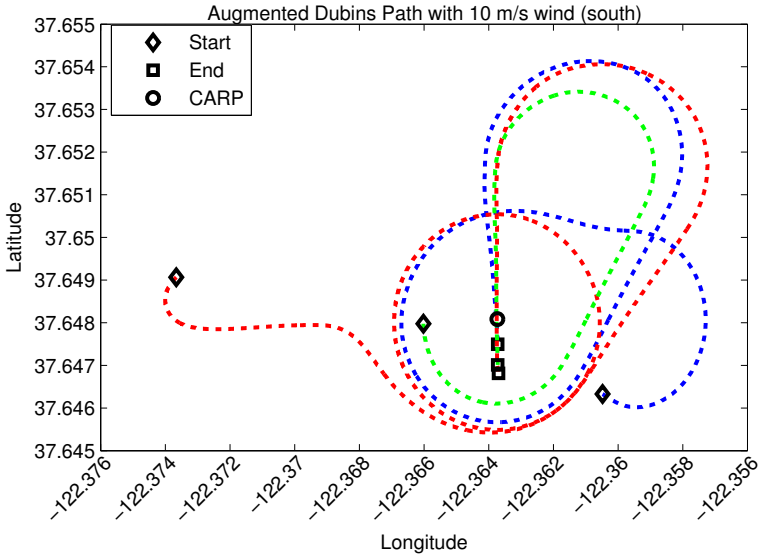


Figure 5.11.: UAV path tracking of the ADP with 10 m/s wind from south.

Although the previous figures display a seemingly good accuracy of the end configuration in the ADP, it was necessary to perform a long time simulation to determine whether this result was consistent. It can be seen from Fig. 5.12 that the deviation in x is approximately the same as previously seen in Fig. 5.7. This further substantiates the point that this error is mainly caused by the limit in logging frequency of the aircraft position.

More importantly, however, is the deviation along the y-axis. Throughout this test, the aircraft only deviated from the path by no more than 0.5 m. This is a significant improvement compared to the results of the SLA simulation. In total it can be seen that the horizontal deviation from the desired position is primarily affected by the positional logging frequency.

In the same way that the positional deviation is improved by using the ADP, so is the angular deviation. This result can be seen in Fig. 5.13, which displays an estimated normal distribution of the angular deviation based on measurements from the long time test of the ADP. It can clearly be seen that the angle deviated by less than one degree, which is a remarkable improvement from the performance of the SLA.

Much like with the SLA, however, the ADP also possessed a slight bias to one side. As before, this is caused by the way the Piccolo SL Autopilot interprets and executes a transition from the current position and towards the next waypoint.

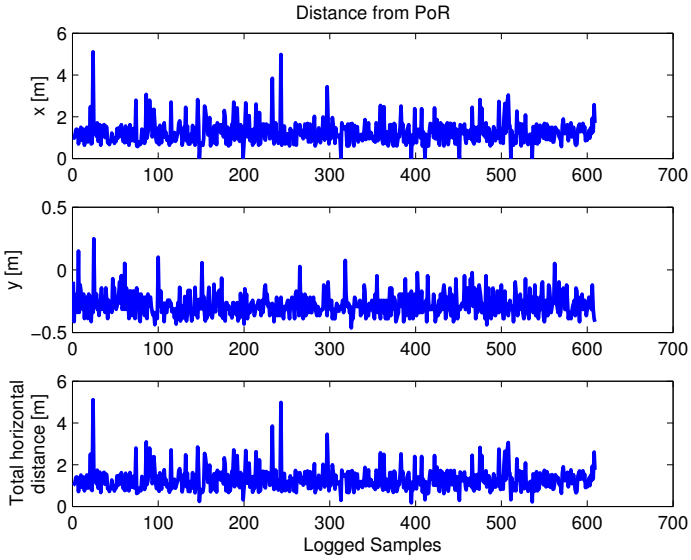


Figure 5.12.: Long time testing of the deviation from the CARP with the ADP.

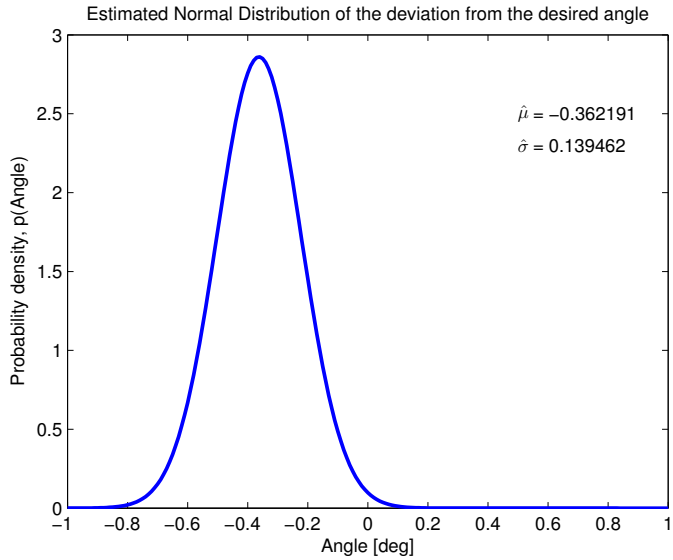


Figure 5.13.: Estimated Normal distribution of the deviation from the desired angle based on the measurements from the ADP.

5.1.3. Free Fall Simulation

There are several inaccuracies in this method of precision airdrop that may affect the final outcome, i.e. the actual point of impact. As mentioned, the physical properties of the released beacon affects the trajectory of the free fall. Another inaccuracy is caused by the air density, which here was assumed to be constant. However, the effects of these sources of error will be neglected in this analysis. Instead, the focus will be put on the effects of the wind and the initial position and direction of the free fall.

As mentioned before, the wind can be estimated as a vector based on the average wind. However, the wind does not act as a uniform vector field. On the contrary, the wind depends on the geographical topography and can vary largely from one altitude to another. Moreover, random wind gusts provides an even larger uncertainty in the wind vector. Thus, in an attempt to estimate the wind uncertainty, it was decided to utilize Matlab functionality to normally distribute the wind in both magnitude and direction. This probability distribution, displayed in Fig. 5.14, assumes that the wind magnitude is normally distributed with $\mu_{magnitude} = 5$ and $\sigma_{magnitude} = 0.48$. These values are based on the assumption that the wind magnitude varies by 50 %. Moreover, the wind angle is normally distributed with $\mu_{direction} = \frac{5}{4}\pi\,(= 225^{\text{o}})$ and $\sigma_{direction} = 0.30$, which is based on the assumption that the wind may vary by up to $\frac{\pi}{2}$ (or 90^{o}) in both directions.

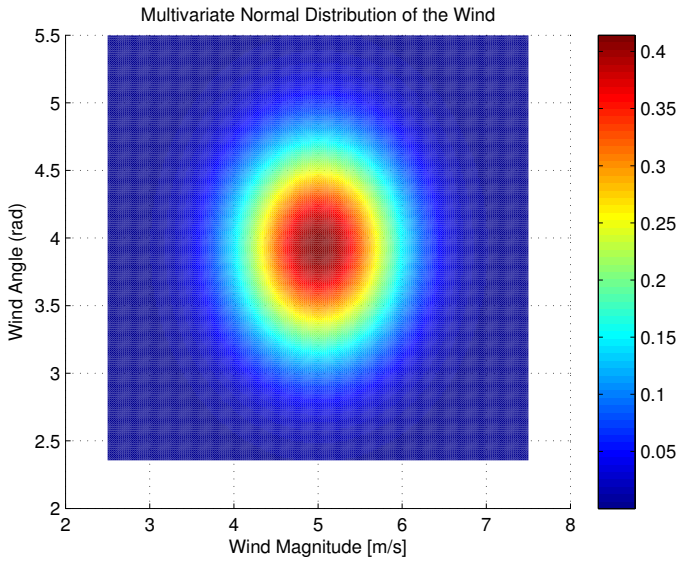


Figure 5.14.: A multivariate normal distribution of the wind magnitude and direction.

5.1.3.1. Accurate Release

In the first simulation the beacon is released from the origin and in the direction opposite to that of the wind, i.e. $\frac{\pi}{4}$ (or 45°). This is also the point and direction used in calculating the expected point of impact. As such, it can be said that the beacon is released in a perfect manner. However, assuming that the wind can be estimated by the previously mentioned multivariate normal probability distribution, the point of impact is expected to vary accordingly. Throughout these simulations, based on the release altitude, the calculated point of impact can be seen in Tab. 5.1.

Fig. 5.15 displays the likely area of impact as a probability distribution based on the aforementioned criteria. It can here be seen that given a release altitude of 30 m, the expected area of impact is relatively small,

even with the uncertain wind disturbance. Releasing the beacon from 50 m, on the other hand, increased the likely area of impact as shown in Fig. 5.16. And, as expected, if the release altitude is increased to 100 m, as shown in Fig. 5.17, the size of the likely area of impact is increased drastically. This result reveals that the accuracy of this method of precision airdrop is highly correlated with the altitude at which it was released.

Altitude of Release [m]	Expected Point of Impact, $(x, y)_{impact}$
30	(28.5, 28.5)
50	(32.8, 32.8)
100	(36.4, 36.4)

Table 5.1.: The calculated point of impact based on the release altitude and an assumed 5 m/s wind.

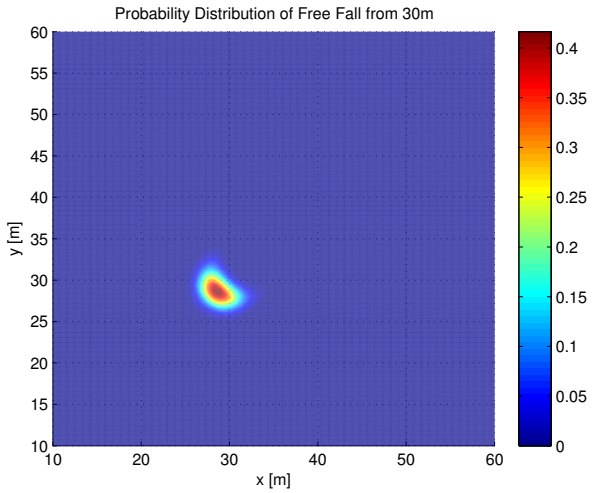


Figure 5.15.: The probability distribution of the area of impact when the beacon is released from 30 m given a perfect release configuration.

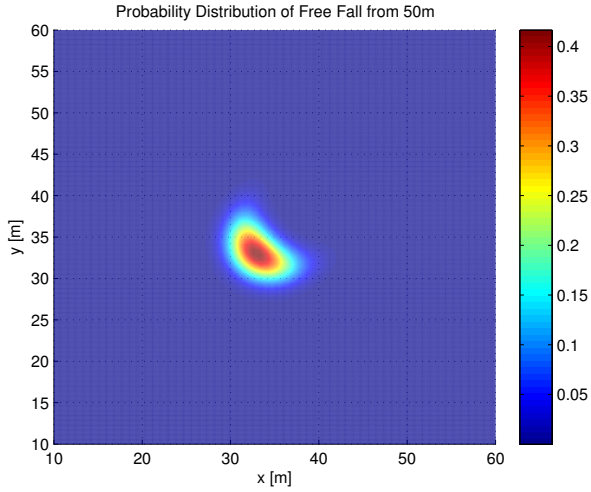


Figure 5.16.: The probability distribution of the area of impact when the beacon is released from 50 m given a perfect release configuration.

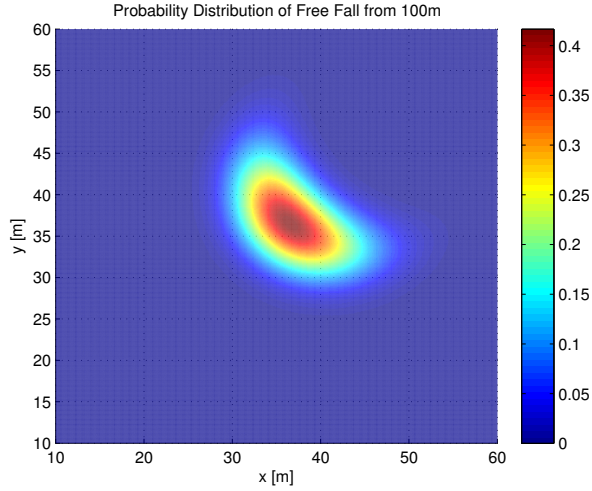


Figure 5.17.: The probability distribution of the area of impact when the beacon is released from 100 m given a perfect release configuration.

5.1.3.2. Release Configuration based on the Straight Line Approach

The HIL simulations of the SLA and the ADP revealed that releasing the beacon from a perfect vehicle configuration is very unlikely. Especially when the SLA is to be considered, as it revealed a severe lack of ability to accurately achieve the desired release configuration. Consequently, this behavior affects the expected area of impact. As such it was decided to combine the previously obtained long time data, displayed in Fig. 5.7 and Fig. 5.8, with the estimated area of impact based on the wind disturbance. This way the two main contributors to error were combined and could further provide a picture of how the likely area of impact would appear.

As displayed in Fig. 5.18, the SLA based release configuration clearly displaced the expected area of impact to a different location compared to a perfect release configuration. In Fig. 5.19 the free fall simulation based on the actual release configuration, from using the SLA, has been subtracted from the free fall simulation based on the ideal release configuration, and consequently created a plot that reveals the difference between the two. Here the aforementioned configuration bias can be seen to clearly influence the area of impact, even with a release altitude of only 30 m.

Moreover, when considering Fig. 5.20 and Fig. 5.22, it becomes evident that the altitude of release impacts the size of the area of impact. Further, Fig. 5.21 and Fig. 5.23, displays how the ideal and actual area of impact differs with an increasing release altitude. However, due to the consistency of the data collected from the long time simulation, this method does not increase the expected area of impact significantly.

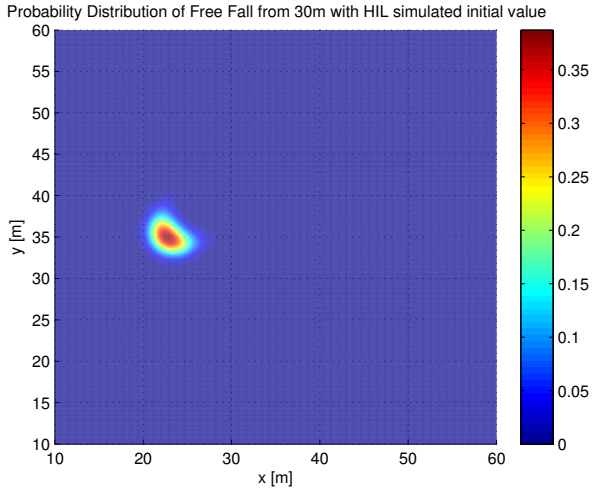


Figure 5.18.: The probability distribution of the area of impact when the beacon is released from 30 m based on the SLA release configuration.

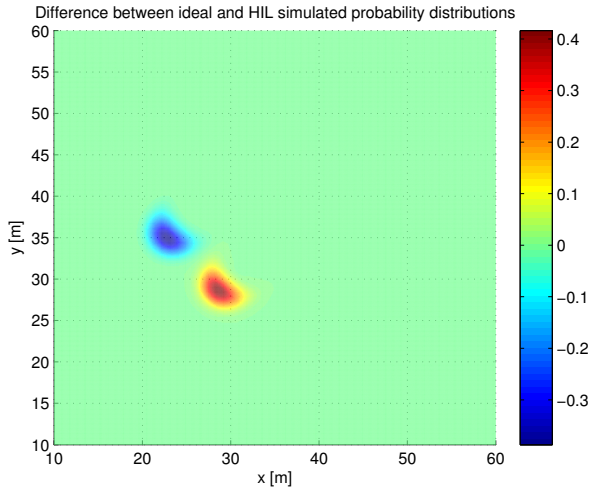


Figure 5.19.: The difference between the ideal release configuration and the SLA release configuration from 30 m.

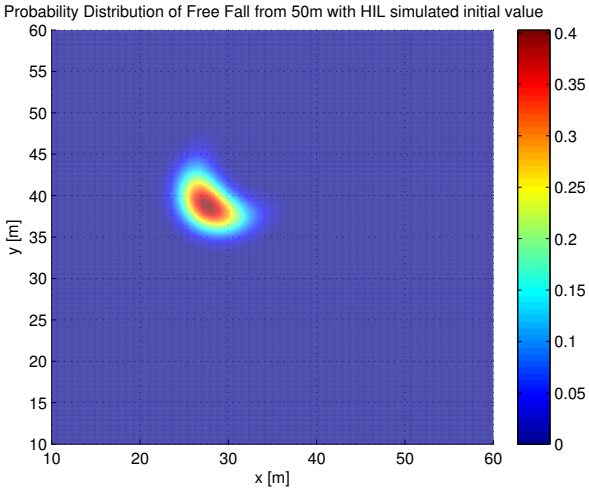


Figure 5.20.: The probability distribution of the area of impact when the beacon is released from 50 m based on the SLA release configuration.

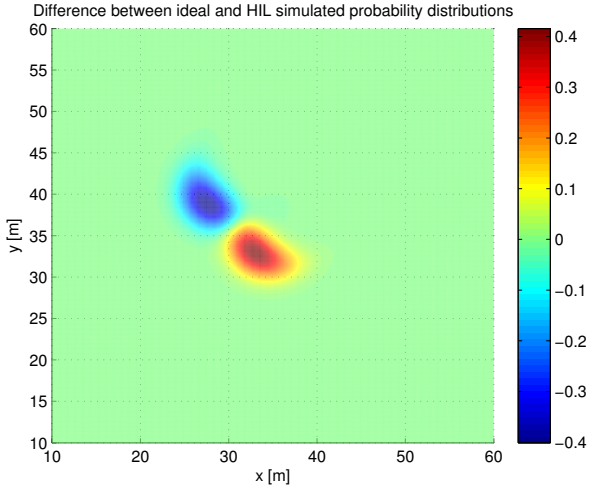


Figure 5.21.: The difference between the ideal release configuration and the SLA release configuration from 50 m.

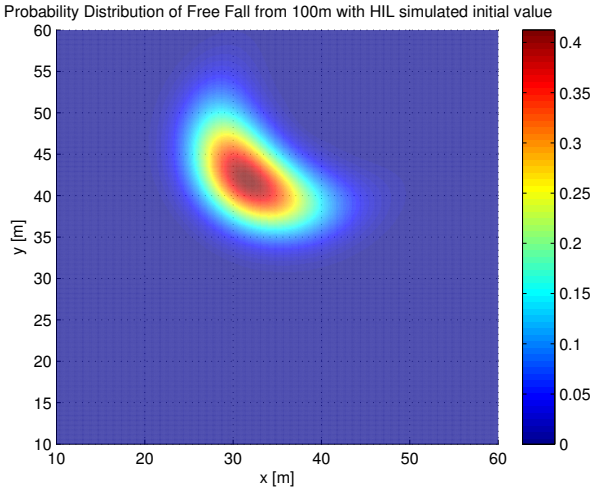


Figure 5.22.: The probability distribution of the area of impact when the beacon is released from 100 m based on the SLA release configuration.

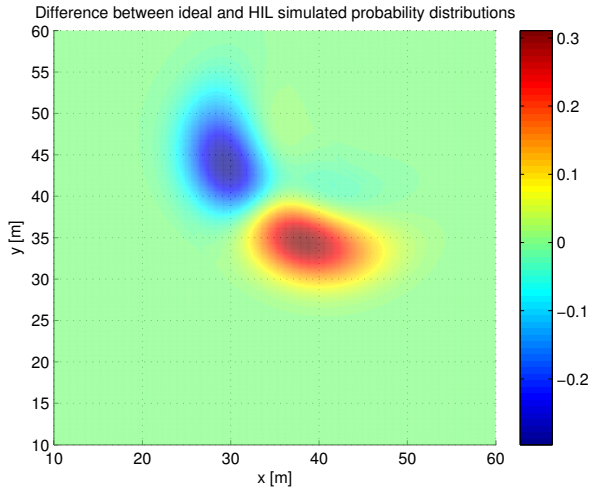


Figure 5.23.: The difference between the ideal release configuration and the SLA release configuration from 100 m.

5.1.3.3. Release Configuration based on the Augmented Dubins Path

In sec. 5.1.2.2 it was shown that the ADP displayed a high degree of ability to track the desired path, and further it managed to achieve the desired end configuration. By combining the long time simulation of the ADP, along with the estimated wind disturbance, it was possible to obtain an estimate of how the area of impact would appear if the ADP was utilized.

Due to the high achievable accuracy of the release configuration, based on the ADP, it can be seen in Fig. 5.24 that the expected area of impact is very similar to that found when assuming a perfect initial configuration, shown in Fig. 5.15. In fact, Fig. 5.25 reveals that the ideal and the actual area of impact differs only by a few meters. Moreover, it can be seen that the actual area of impact is primarily shifted diagonally further away. This happens because the aircraft releases the beacon after crossing the CARP. And as it was shown in sec. 5.1.2.2, the primary positional error source of the ADP was along the x-axis in the BODY frame, which was caused by the limited logging frequency.

Furthermore, like with the SLA, this algorithm also reveals a high degree of consistency. Consequently the area of impact does not change noticeably in shape or size. However, as previously explained, and as shown in Fig. 5.26 and Fig. 5.28, the altitude plays a significant role in determining the possible area of impact. Both Fig. 5.27 and Fig. 5.29 reveals how the deviation from the ideal area of impact increases with increasing altitude. As such, it can be seen that even with the ADP, the distance between the expected and the actual point of impact increases drastically with a release from a higher altitude.

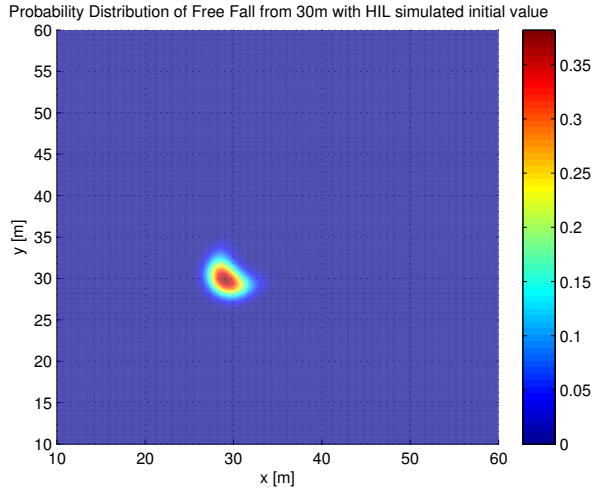


Figure 5.24.: The probability distribution of the area of impact when the beacon is released from 30 m based on the ADP release configuration.

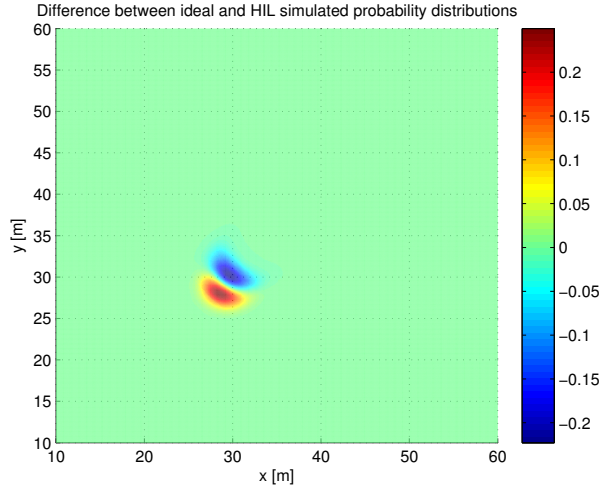


Figure 5.25.: The difference between the ideal release configuration and the ADP release configuration from 30 m.

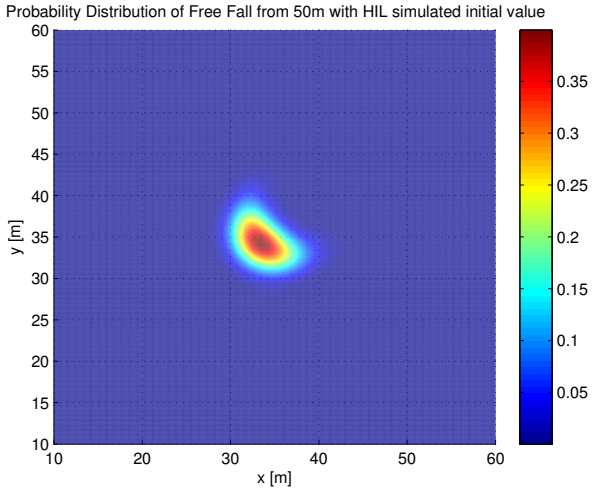


Figure 5.26.: The probability distribution of the area of impact when the beacon is released from 50 m based on the ADP release configuration.

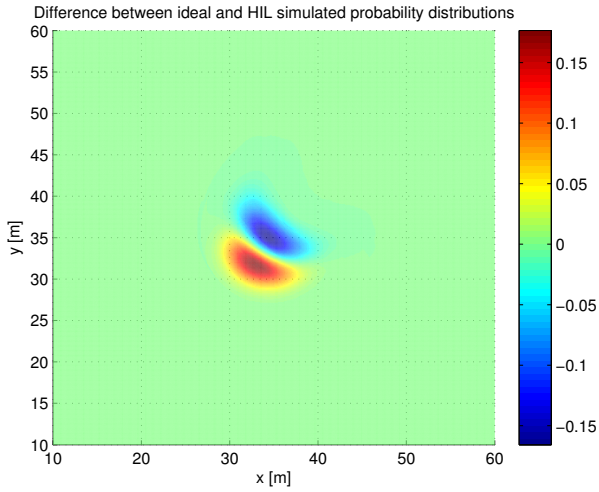


Figure 5.27.: The difference between the ideal release configuration and the ADP release configuration from 50 m.

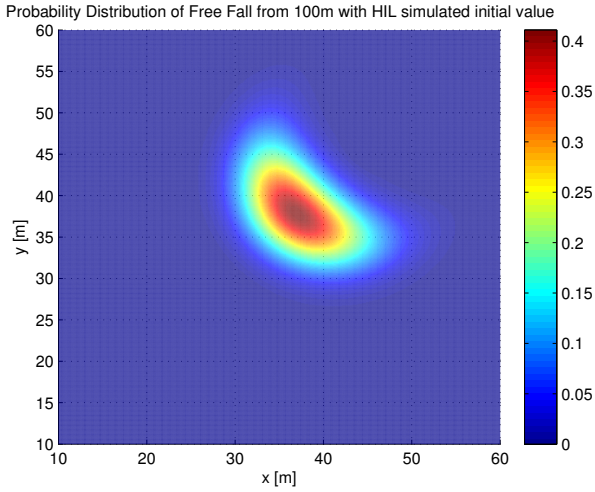


Figure 5.28.: The probability distribution of the area of impact when the beacon is released from 100 m based on the ADP release configuration.

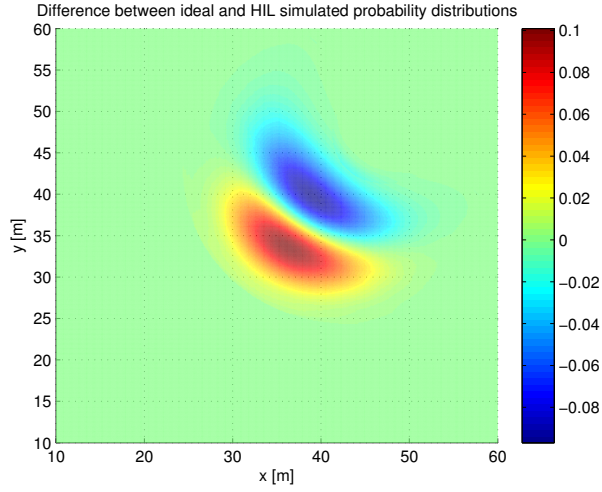


Figure 5.29.: The difference between the ideal release configuration and the ADP release configuration from 50 m.

5.2. Testing with the Penguin B on Agdenes Airstrip

Although the data provided by the software simulations could explain how the aircraft would, most likely, behave, it was naturally desirable to test the system in the real world. This meant that the implemented software had to be set up to work with the payload, which was presented in sec. 4.1. Furthermore, to ensure that the payload worked as intended, the full payload system was connected to the HIL simulator and successfully tested.

5.2.1. Ground Testing

During the first test on Agdenes it was necessary to ensure that the release mechanism functioned as intended. However, due to unforeseen technical difficulties out of control for the project team, the aircraft could not be flown, and consequently the test had to be performed on the ground. Regardless, because the algorithm simply checked the distance in the horizontal plane, this would still provide a useful result.

The Penguin B was taxiing on the ground and approached a given coordinate in order to activate the release mechanism. Due to the low achievable speed on the ground, it was sufficient to set the circle radius to five meters and observe what happened when the aircraft was within this perimeter. Upon entering the circle, the release mechanism activated and the test was deemed successful.

5.2.2. In-Air Testing

Although airborne tests were highly desired, it became clear late in the project that this was not possible to accomplish due to a hardware failure on the aircraft. Thus, a suggestion to change UAV platform to the smaller X8 was proposed and considered. By changing aircraft it would be possible to test the final system in the air and obtain some real data on how the algorithm would perform and how accurately the beacon would land relative to the given target. The transition from the Penguin B to the X8 would, however, potentially cause some problems.

The X8 would be able to utilize the previously developed DUNE software. However, because the X8 utilizes a flight control unit known as ArduPilot instead of the previously introduced Piccolo SL, the existing code would require some modifications. Additionally, it would be necessary to alter and install the payload, along with the release mechanism, on the X8 aircraft.

Lastly, by changing to ArduPilot it would not be possible to obtain simulated data from the same HIL simulator. Consequently, if the transition from the Penguin B to the X8 was unsuccessful, there would not be any results.

After careful consideration it was decided that the change of aircraft would not be feasible this late into the project and, as such, the continued use of the Penguin B was chosen.

6. Discussion

6.1. Release Criteria

The matter of choosing the most suitable release criterion was important as it clearly affected the accuracy of the release. The circle criterion appeared to be the most accurate as it could release the beacon both before and after passing the CARP. As such the position of release would evidently be closer to the CARP compared to the case where it would only release the beacon after passing the CARP, as was the case with the halfplane criterion.

Nevertheless, in order to ensure that the accuracy of the circle criterion was preserved, it was necessary to set the radius of the circle very small. However, as was previously mentioned, the utilization of a very small circle may cause the aircraft to never reach the CARP at all. When considering this, it is evident that the halfplane criterion could provide a better accuracy and reliability. On the other hand, by combining the two criteria, with a sufficiently large radius, it may be possible to achieve both an accurate and reliable result. Flight testing would provide a way to fine-tune settings such as the size of the circle.

6.2. Path Control

Along with the release criterion, it was also important to control the UAV in the most optimal way in order to approach the CARP with the most optimal configuration. It was initially desired to control the UAV by utilizing heading control. However, it was explained in sec. 3.4.3, that it would not be possible to obtain an accurate measurement of the heading. Consequently, it was necessary to utilize waypoints as the control input.

Two different approaches, which could both be implemented by the use of waypoints, were introduced. The first approach, namely the Straight Line Approach, consisted simply of two waypoints aligned in the desired direction of flight. The results from the simulations of this method revealed that this method struggled to approach the final configuration in a desired manner. In fact, a consistent and significant bias, both in the position and angle, was observed. Additionally, it was seen that the end configuration of the SLA appeared to depend, to some degree, on the initial configuration. However, the SLA transitioned from the initial position to the terminal position in an efficient manner.

The Augmented Dubins Path, on the other hand, consisted of several waypoints, which combined both loitering and straight line flying. This way it was possible to follow a somewhat predetermined path, which would continuously guide the aircraft in a more optimal way. It was evident from the results of the simulations that this method followed the predetermined path in a highly reliable manner. Moreover, even with the addition of the wind disturbance, this method was capable of adapting to this disturbance by altering the properties of the path. Consequently, this method managed to consistently achieve

the desired release configuration with only a minor bias in both the position and angle, regardless of the initial configuration.

6.3. Free Fall

It has previously been explained that the decided method of precision airdrop, known as free fall, possesses several sources of error. Of these errors, two of them were considered to have a larger impact on the final result, namely the unpredictable wind and the actual release configuration. Due to the high uncertainty in the behavior of the wind, caused by different geographical topography, it was decided to estimate the wind uncertainty as a multivariate normal distribution based on the magnitude and direction of the wind. The resulting probability density function could then be used to simulate a free fall under unpredictable wind conditions.

This simulation revealed that, with an ideal release of the beacon, the actual point of impact would, in fact, be within a small proximity of the desired target. However, as the altitude of release increased, it was easy to observe that the likely area of impact increased along with it. On the other hand, should the wind prove to be more predictable and uniform, which could be the case in open waters with less topography, the likely area of impact may be smaller, and thus the precision airdrop may be more accurate than visualized here.

As the latter simulations displayed, it was not possible to assume a perfect release configuration. In fact, the biased release configuration based on the SLA was of such a magnitude that the resulting area of impact was changed significantly. The ADP based release configuration, on the other hand, displayed a remarkable resemblance to that

of the ideal one. Consequently, the actual area of impact was much closer to the ideal area of impact. This result clearly visualizes how important it is to utilize a good control algorithm, and consequently achieve an optimal release configuration.

7. Conclusion

The goal of this project was to achieve an accurate and precise airdrop from the fixed-wing UAV Penguin B. Based on the provided framework and the core assumptions, an analysis was conducted and the method of releasing the beacon in free fall was deemed the most appropriate. This thesis further described how the point of release, i.e. the CARP, was calculated, and how the aircraft should approach this point. It was further concluded that by combining the circle and the halfplane criteria, it would be possible to achieve an accurate and reliable release.

Two paths, the SLA and the ADP, which both guided the aircraft to the desired configuration, were introduced. Furthermore, an evaluation of the control input to the Piccolo SL Autopilot was conducted which concluded that the use of waypoints was the most suitable for this project.

The entire system was implemented, both on the software and on the hardware side, according to the desired specifications. Finally the system was tested by the use of the HIL simulator in addition to some limited testing in the field. The simulations revealed that the accuracy of the free fall method decreased drastically with an increasing release altitude. However, under the assumption that the aircraft is able to operate at an altitude of 30 m, the simulations revealed a high degree of achievable accuracy in the free fall method. It was also shown that

the release configuration played a significant role in the actual point of impact, which further implied that the path planning and the path tracking are of huge importance in this system.

In conclusion, based on the results from simulations and the considered assumptions, the final system, using the ADP to approach the CARP, is able to achieve a high degree of accuracy in the deployment of the beacon.

8. Further Work

Based on the experience from the conducted project, this chapter introduces a few suggestions for changes and ideas for future work.

8.1. Improved Aircraft Control

The performance of the ADP, as presented in this thesis, was able to achieve a release configuration which was very close to the desired configuration. However, by adding a way of accurately measuring the heading, it is possible to utilize the heading input control, which may result in an even more optimal release configuration. The heading control may be performed by utilizing the theory behind the MPC or the CLF. As sec. 5.1.3 presented, the accuracy of the free fall method depends to a large degree on the release configuration. By further optimizing the control of the aircraft, it is possible to achieve a more accurate release configuration, and consequently a more accurate result from the free fall method may be achieved.

It may also be relevant to increase the control of the aircraft from the horizontal plane to include the vertical axis. Such a change may allow for the aircraft to quickly descend into the lowest acceptable altitude, release the beacon, and then ascend to a safer operational altitude. As was shown in this thesis, the altitude plays a significant role in the

accuracy of the free fall, and consequently this solution may increase that accuracy.

8.2. UAV Platform

It was previously mentioned in this thesis that it was possible to change UAV from the Penguin B to the smaller, lighter and less expensive X8. The transition between the two was deemed to be possible, however, it was likely not a straight forward task. Nevertheless, there may be several benefits of changing UAVs. At the beginning of this project, the Penguin B was still in the start-up phase and consequently there was not enough experience with this aircraft. The X8, on the other hand, has previously been thoroughly tested. Thus, the X8 provides a more readily available UAV platform. This further leads to the possibility of testing both the methods implemented as a part of this thesis, as well as other ways of accomplishing the precision airdrop task.

8.3. User Interface

Although desirable, a user interface was not implemented in this project. However, by utilizing the functionality of Neptus it should be possible to create a user interface in which the target position can be set on a map, rather than typed into the current Ground Station task.

Bibliography

[1] S. Tiffin, I. Turnbull, T. Sylvestre, and J. Acevedo, "21st iahr international symposium on ice," 2012.

[2] R. J. Ducote and R. J. Speelman, "U. s. air force concepts for accurate delivery of equipment and supplies," in *Aerodynamic Deceleration Systems Conference*, vol. 4, July-August 1966.

[3] B. Vik, "Integrated satellite and inertial navigation systems," *Department of Engineering Cybernetics, NTNU*, 2012.

[4] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[5] S. E. d. N. K. Hrabar, *Vision-based 3D navigation for an autonomous helicopter*. PhD thesis, Citeseer, 2006.

[6] W. Ren and R. W. Beard, "Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints," *Control Systems Technology, IEEE Transactions on*, vol. 12, no. 5, pp. 706–716, 2004.

[7] L. Singh and J. Fuller, "Trajectory generation for a uav in urban terrain, using nonlinear mpc," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 3, pp. 2301–2308, IEEE, 2001.

[8] S. Leven, J.-C. Zufferey, and D. Floreano, "A minimalist control strategy for small uavs," in *Intelligent Robots and Systems, 2009.*

IROS 2009. IEEE/RSJ International Conference on, pp. 2873–2878, IEEE, 2009.

[9] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.

[10] "Federal aviation administration definitions." `http://www.ecfr.gov/cgi-bin/text-idx?SID=a437b47edd5dfc18ae6693d1f6c1ee7a&node=14:1.0.1.1.1.0.1.1&rgn=div8`, May 2014. Accessed on: 2014.05.21.

[11] UAV Factory USA LLC, 50 South Buckhout Street, Irvington, NY 10533 USA, *Penguin B Datasheet*, v2.0 ed.

[12] "Uav factory - penguin b." `http://www.uavfactory.com/product/46`.

[13] UTC Aerospace Systems, 202 Wasco Loop, Suite 103, Hood River, OR 97031 USA, *Piccolo SL Datasheet*, 2014.

[14] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The lsts toolchain for networked vehicle systems," in *OCEANS-Bergen, 2013 MTS/IEEE*, pp. 1–9, IEEE, 2013.

[15] P. b. c. Williams and P. b. Trivailo, "Cable-supported sliding payload deployment from a circling fixed-wing aircraft," *Journal of Aircraft*, vol. 43, no. 5, pp. 1567–1570, 2006. cited By (since 1996)2.

[16] P. Williams and P. Trivailo, "Dynamics of circularly towed aerial cable systems, part i: Optimal configurations and their stability," *Journal of guidance, control, and dynamics*, vol. 30, no. 3, pp. 753–765, 2007.

[17] M. R. Wuest and R. J. Benney, "Precision airdrop (largage de precision)," *Flight Test Techniques Series*, vol. 24, December 2005.

[18] J. M. Cimbala and Y. A. Çengel, *Essentials of fluid mechanics: fundamentals and applications*. McGraw-Hill Higher Education, 2008.

[19] G. Parker, "Projectile motion with air resistance quadratic in the speed," *Am. J. Phys*, vol. 45, no. 7, pp. 606–610, 1977.

[20] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[21] B. Foss and T. A. N. Heirung, "Merging optimization and control." September 2013.

[22] H. K. Khalil and J. Grizzle, *Nonlinear systems*, vol. 3. Prentice hall Upper Saddle River, 2002.

[23] J. W. Curtis and R. W. Beard, "Satisficing: A new approach to constructive nonlinear control," *Automatic Control, IEEE Transactions on*, vol. 49, no. 7, pp. 1090–1102, 2004.

[24] E. Skjong and S. A. Nundal, "Tracking objects with fixed-wing uav using model predictive control and machine vision," Master's thesis, NTNU, June 2014.

[25] S. H. Mathisen, "High precision deployment of wireless ssensor from unmanned aerial vehicles," Master's thesis, NTNU, May 2014.

[26] E-flite, *Servoless Payload Release*, 2013.

[27] Tinder Rocketry, *Peregrine Exhaustless CO2 Ejection System*.

[28] "Servoless payload release." `http://www.horizonhobby.com/products/servoless-payload-release-EFLA405`. Accessed on: 2014.05.25.

[29] "Quanum rtr bomb system." http://www.hobbyking.com/hobbyking/store/uh_viewitem.asp?idproduct=10624&aff=800324. Accessed on: 2014.05.25.

[30] Ubiquiti Networks, Inc., *rocket M Datasheet*.

[31] Ubiquiti Networks, Inc., *NanoStation M Datasheet*.

Nomenclature

ADP	Augmented Dubins Path
AMOS	Autonomous Marine Operations and Systems
CARP	Computed Air Release Point
CAS	Calibrated Airspeed
CG	Center of Gravity
CLF	Control Lyapunov Function
DUNE	Unified Navigation Environment
EAS	Equivalent Airspeed
ECEF	Earth-centered Earth Fixed
FAA	Federal Aviation Administration
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
HMI	Human-Machine Interface

IAS	Indicated Airspeed
IMC	Inter-Module Communication
LAN	Local Area Network
LSL	Left-handed arc - Straight - Left-handed arc (Dubins Path)
LSR	Left-handed arc - Straight - Right-handed arc (Dubins Path)
LSTS	Underwater Systems and Technology Laboratory
MPC	Model Predictive Control
NED	North-East-Down
NTNU	Norwegian University of Science and Technology
ODE	Ordinary Differential Equation
OS	Operating System
PCC	Piccolo Command Center
PWM	Pulse-Width Modulated
RSL	Right-handed arc - Straight - Left-handed arc (Dubins Path)
RSR	Right-handed arc - Straight - Right-handed arc (Dubins Path)
SD	Secure Digital

SIL	Software-in-the-Loop
SLA	Straight Line Approach
TAS	True Airspeed
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
WGS	World Geodetic System
WP	Waypoint

List of Figures

List of Tables

A. Software and Hardware

This chapter briefly introduces the software and hardware that was used throughout this project. Although there are more components that may be mentioned in the hardware table, the idea is primarily to create an overview of the specific parts used in this project.

Product	Purpose
Matlab R2013b	Simulation and data analysis
Cloud Cap Technology - Piccolo Command Center	Piccolo SL GUI
Cloud Cap Technology - Simulator	Piccolo SL Simulator
Eclipse C/C++	C/C++ Integrated Development Environment
LSTS Software Toolchain	Software Framework
Boost C++ Libraries	Additional C++ functionality

Table A.1.: Overview of the used software.

Product	Purpose	Manufacturer
Piccolo SL	Flight Control Unit	Cloud Cap Technology
Penguin B	UAV platform	UAV Factory
Universal Payload Mount	Payload platform	UAV Factory
Servoless Payload Release	Release Mechanism	E-flite
Release Mech. Bar	Attach rel. mech. to Penguin B	Workshop (NTNU)
PandaBoard	On-board computer	pandaboard.org
PWM Circuit Board	Create release PWM signal	Siri Mathisen
Power Supply Circuit Board	Convert 12 V to 5 V	Siri Mathisen
UBEC - 5A 5/6 V	Voltage Regulator	Turnigy
2200 mAh 2S 20C LiPo	Battery to power PandaBoard	Turnigy
Rocket M5	On-board communications unit	Ubiquiti Networks, Inc.
NanoStation M5	Ground communications unit	Ubiquiti Networks, Inc.

Table A.2.: Overview of the used hardware.

B. Software Description

This appendix presents the implemented C++ and DUNE software used in this project. An overview of the code is presented first, followed by a more accurate description of each class or task.

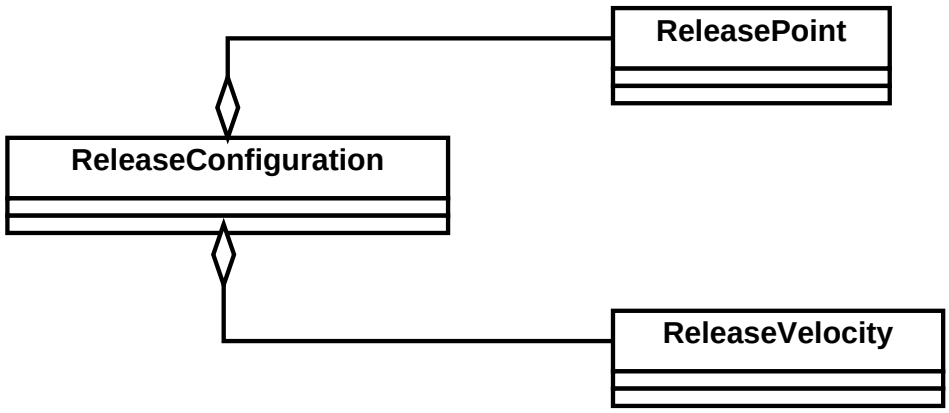


Figure B.1.: ReleaseConfiguration and its relation to ReleaseVelocity and ReleasePoint.

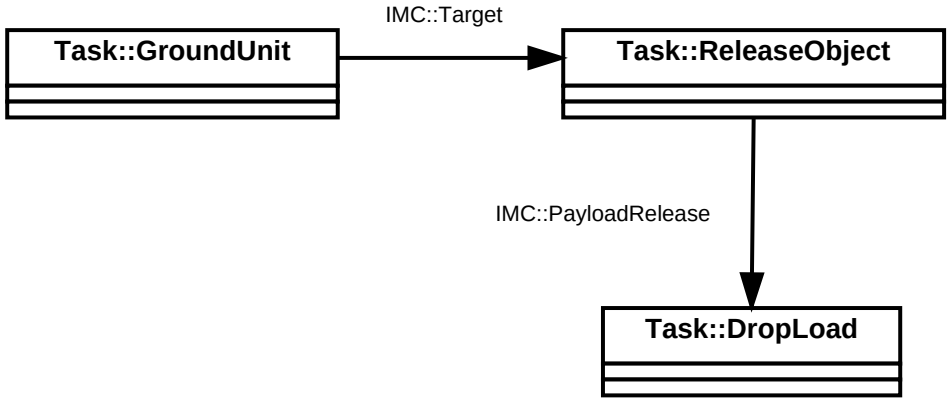


Figure B.2.: Overview of the Object Release test code structure.

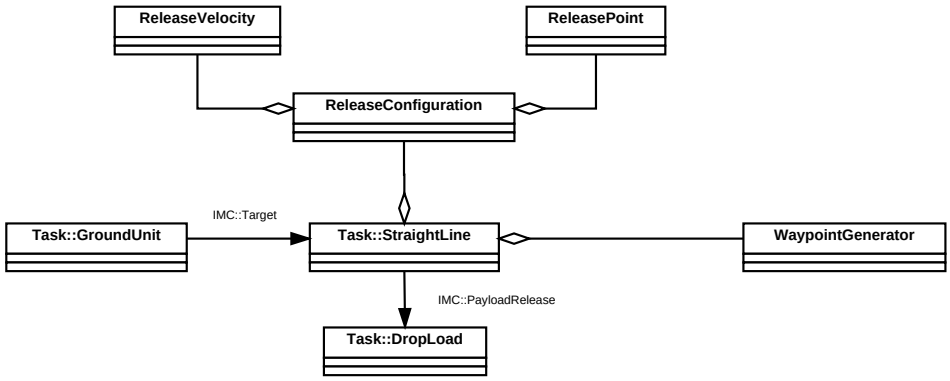


Figure B.3.: Overview of the SLA code structure.

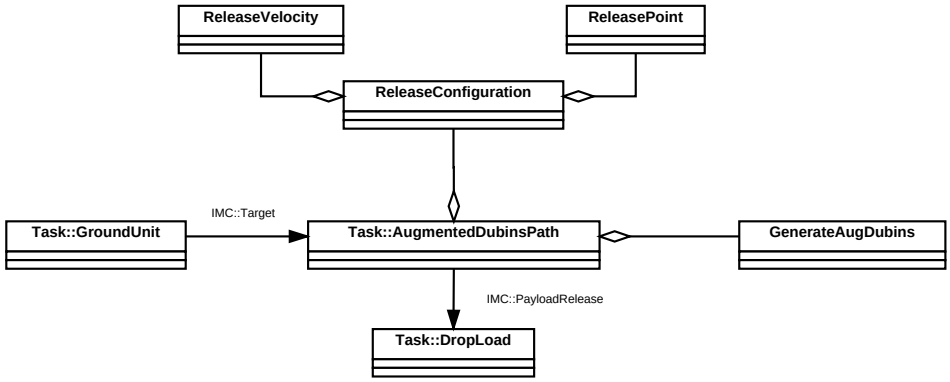


Figure B.4.: Overview of the ADP code structure.

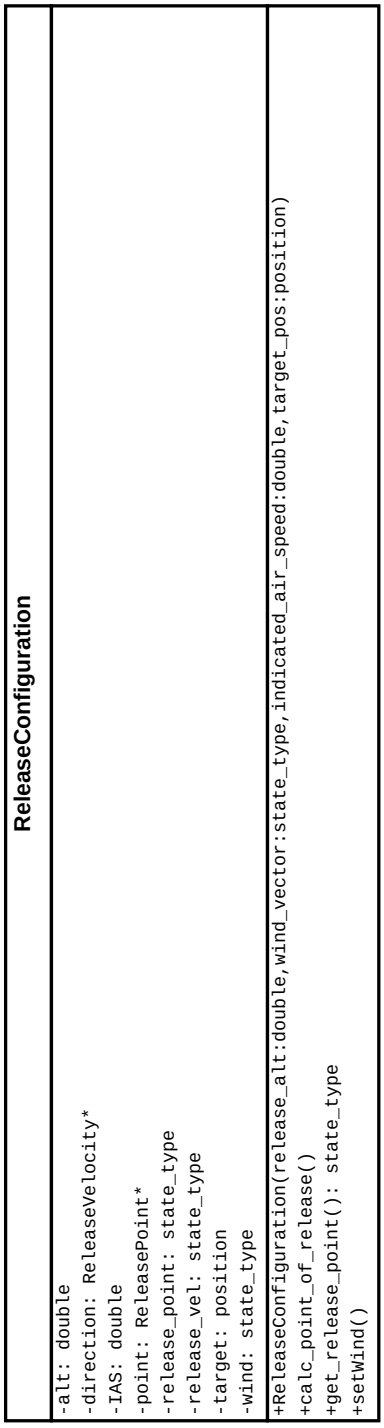


Figure B.5.: ReleaseConfiguration class.

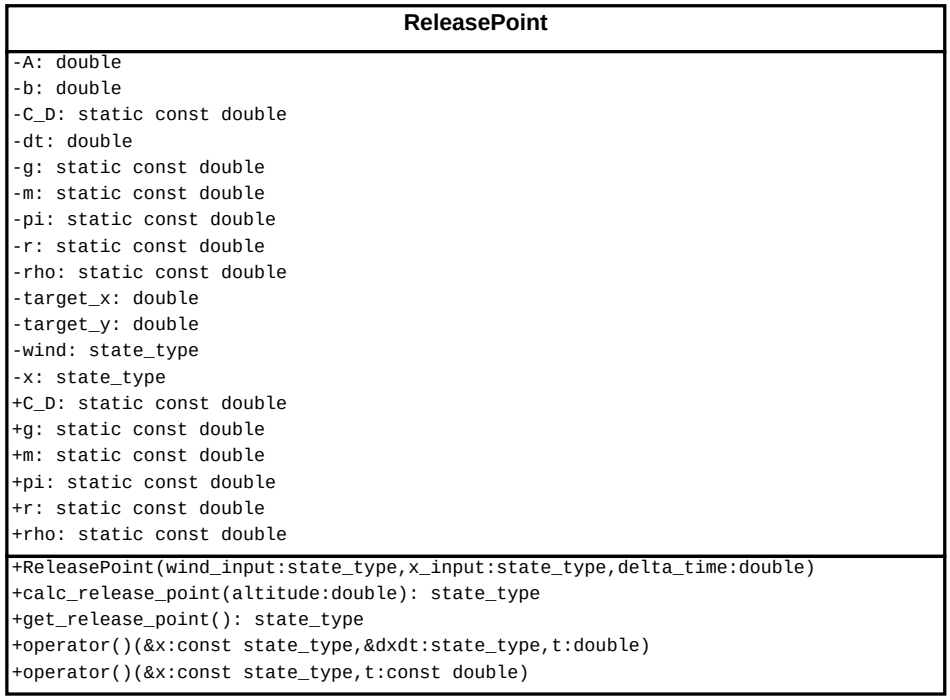


Figure B.6.: ReleasePoint class.

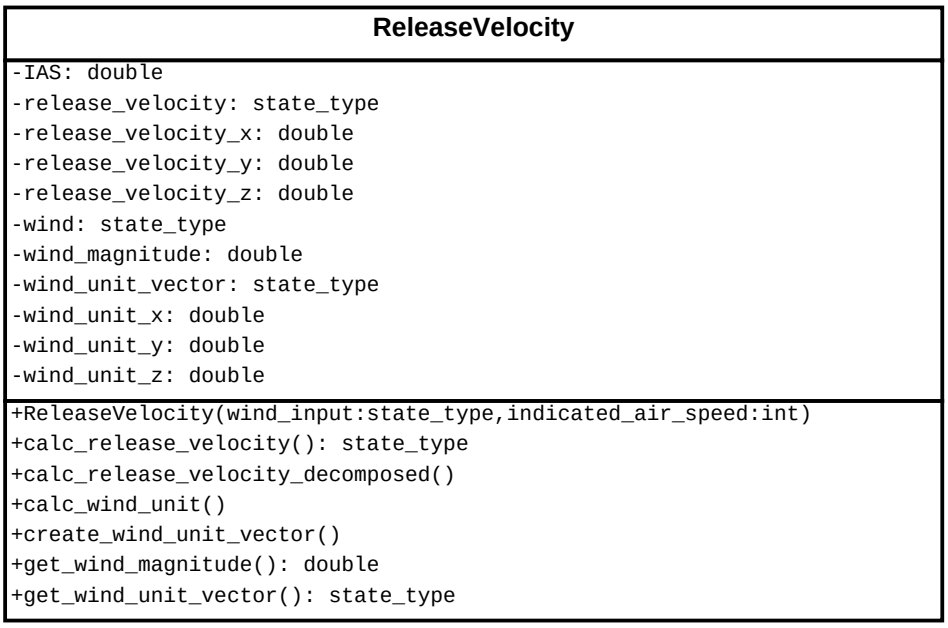


Figure B.7.: ReleaseVelocity class.

Figure B.8.: GenerateAugDubins class.

Figure B.9.: WaypointGenerator class.

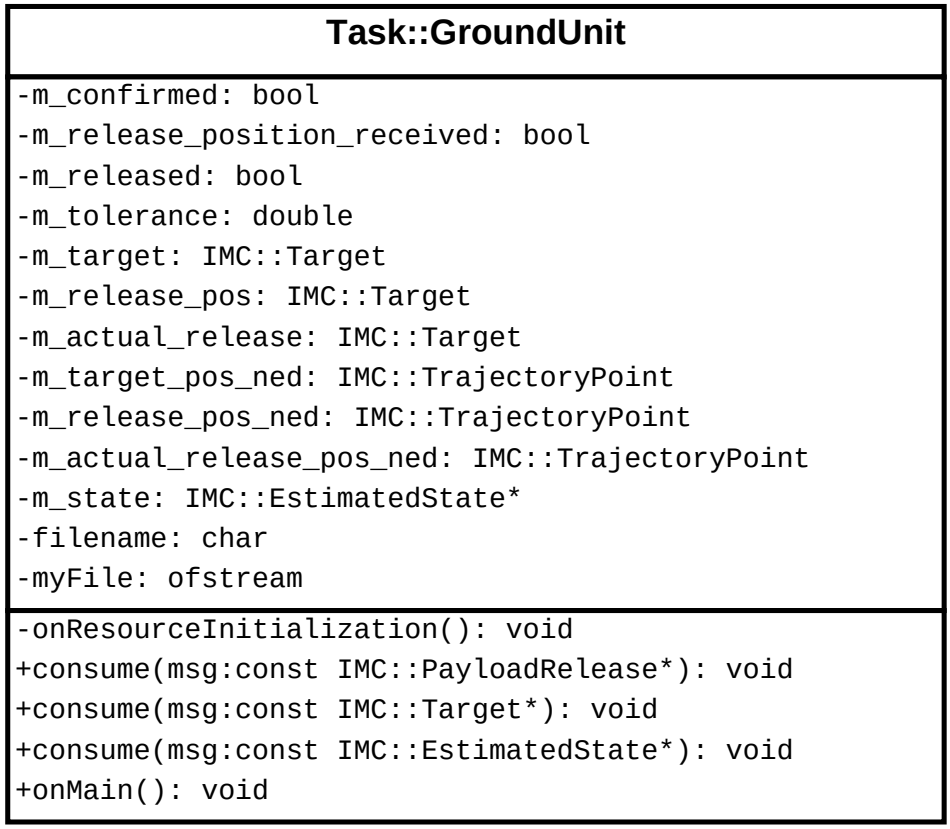


Figure B.10.: GroundUnit DUNE task.

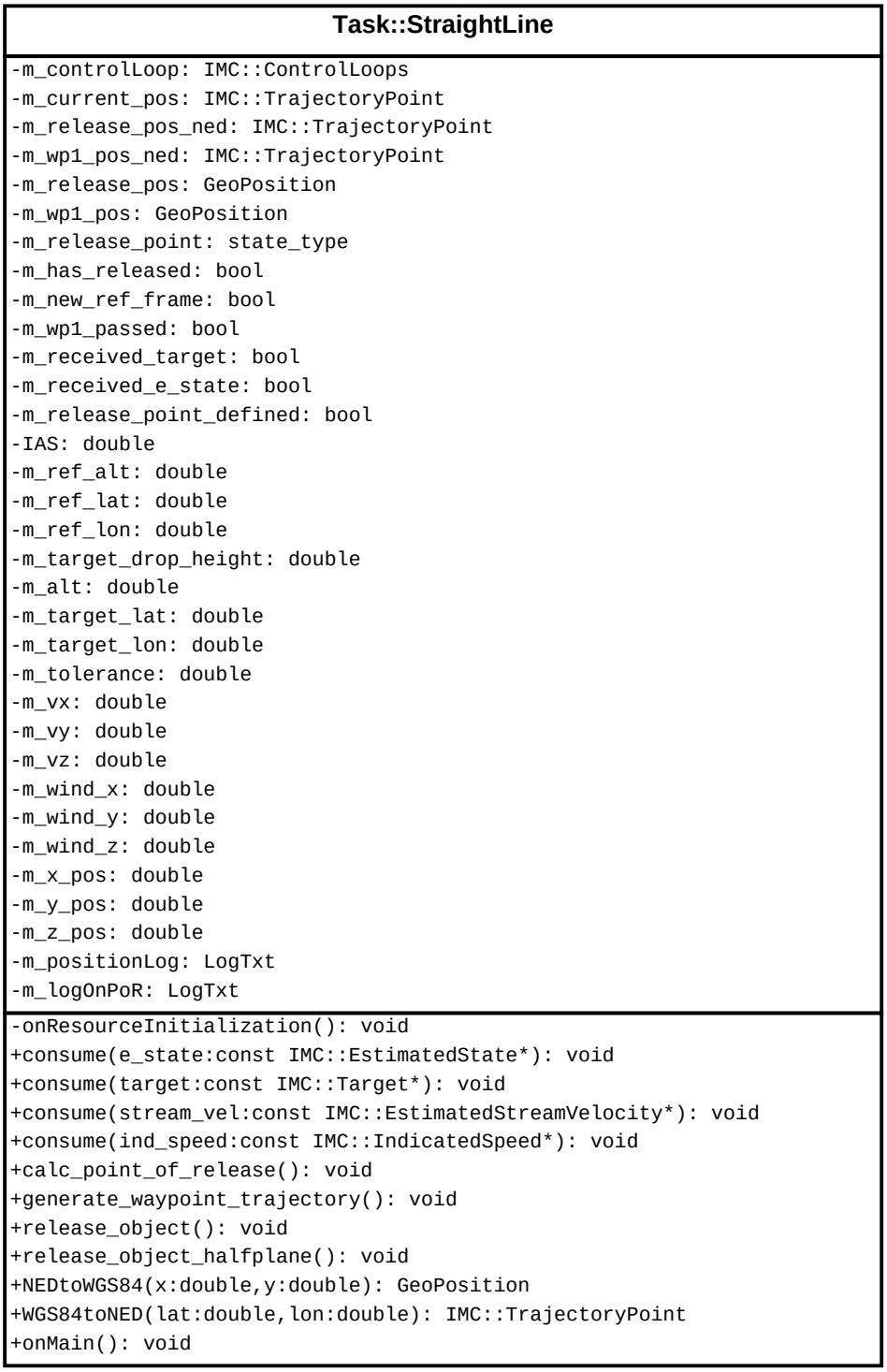


Figure B.11.: StraightLine DUNE task.

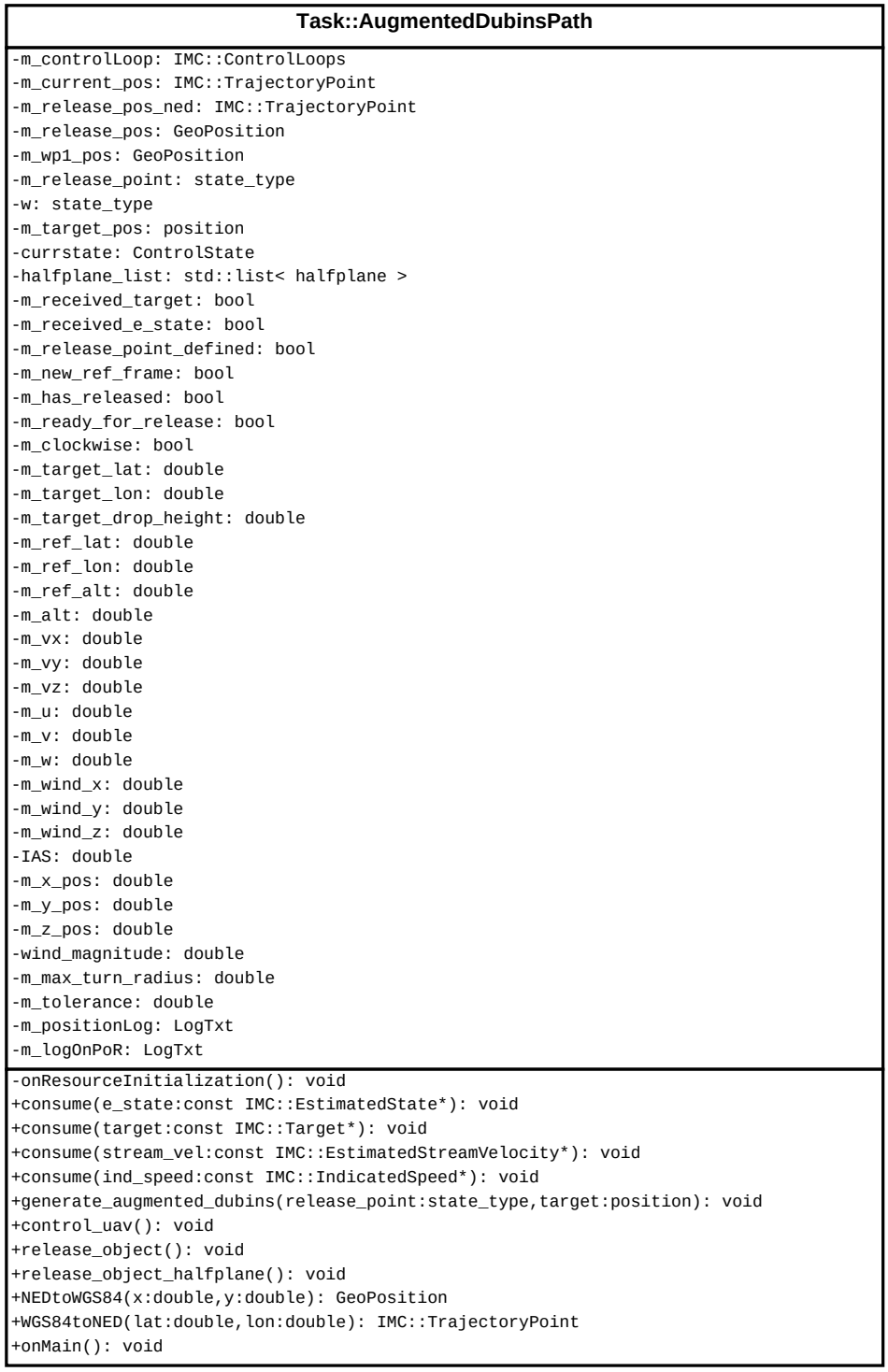


Figure B.12.: AugmentedDubinsPath DUNE task.

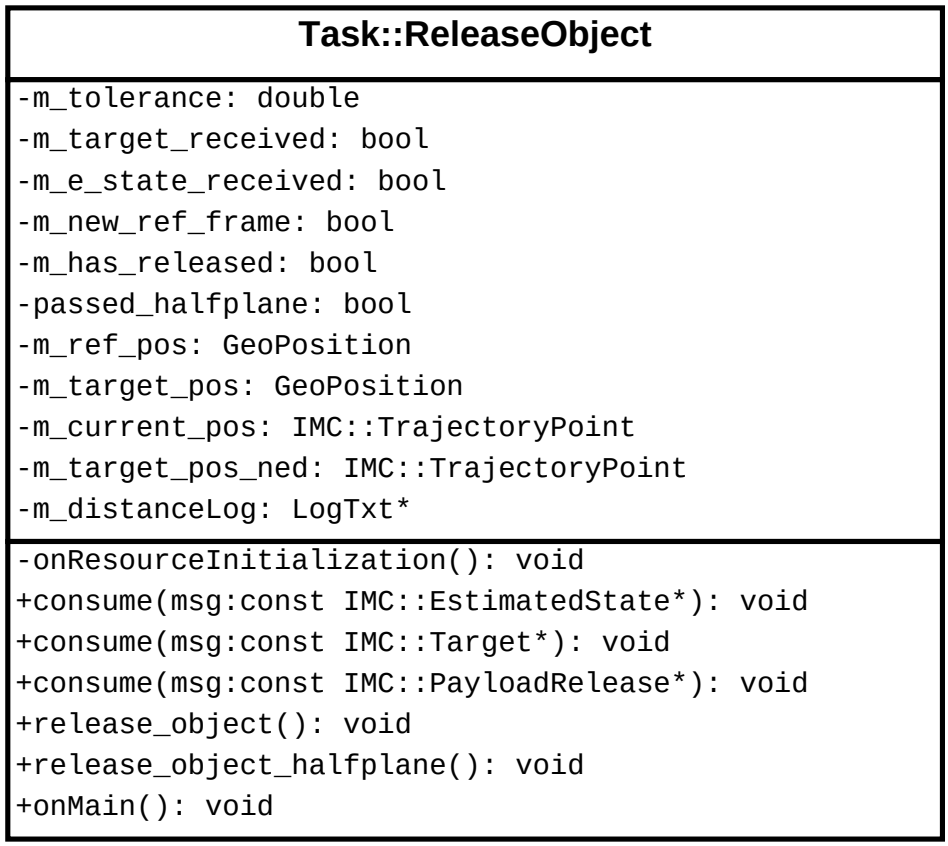


Figure B.13.: ReleaseObject DUNE task.

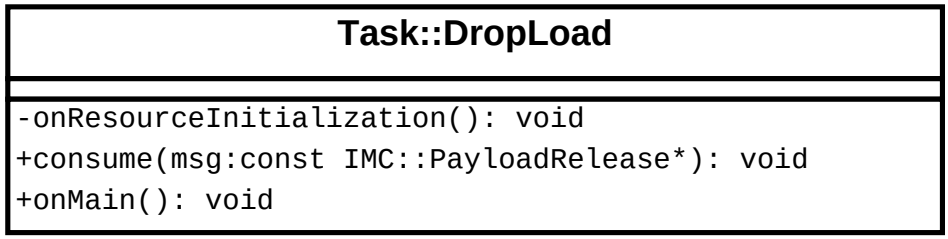


Figure B.14.: DropLoad DUNE task.