Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2020

Robust autonomous landing of fixed-wing UAVs in wind

Tobias Fridén

LINKÖPING
UNIVERSITY

Master of Science Thesis in Electrical Engineering

Robust autonomous landing of fixed-wing UAVs in wind:

Tobias Fridén

LiTH-ISY-EX--YY/NNNN--SE

Supervisor: **Jonatan Olofsson**
ISY, Linköpings universitet
Kristoffer Bergman
ISY, Linköpings universitet
Fredrik Falkman
Airpelago

Examiner: **Daniel Axehill**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

If your thesis is written in English, the primary abstract would go here while the Swedish abstract would be optional.

Contents

1 Introduction

1.1 Background

Unmanned Aerial Vehicles (UAVs) have many different applications, both in commercial usecases such as construction and agriculture, but also in emergency response and personal use. UAVs can be divided into two subclasses, multirotors and fixed-wing UAVs. While both types of UAVs are becoming more and more autonomous through various research efforts, landing of fixed-wing UAVs remains a challenging task which requires manual input from an experienced pilot. For small and light-weight UAVs the presence of wind also acts as a major disturbance which has to be taken into account when planning the landing procedure.

1.2 Scope

1.2.1 Problem formulation

The aim of this thesis is to develop a framework for automatically generating feasible landing procedures for fixed-wing UAVs, under the presence of winds. The landing procedure should be able to take the UAV from an arbitrary initial position and land safely in a predefined landing area while fulfilling physical constraints of the system. This thesis aims to answer the following questions:

1. How can sampling-based motion planning techniques be used to generate landing sequences for fixed-wing UAVs?

2. How can safe landings be guaranteed when taking wind effects into account?

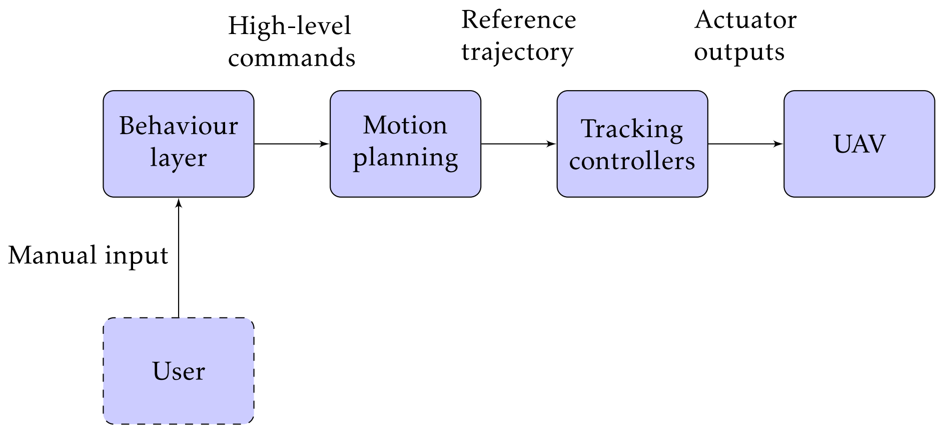


Figure 1.1: *Components of a general autonomous system*

1.2.2 Aim and delimitations

The components of a general autonomous system are illustrated in Figure 1.1. The work in this thesis is mainly focused on the motion planning component. However, the tracking controllers and parameters of the actual UAV have to be taken into account to ensure feasibility of the generated path. Furthermore, the main focus of this thesis is UAVs using the ArduPilot open source autopilot [28].

1.3 Outline

Chapter 2 introduces general concepts regarding fixed-wing UAVs and the kinematic models and controllers studied in this thesis. Chapter 3 defines the wind field and discusses methods to estimate wind. Chapter 4 gives an overview of necessary motion planning theory. In Chapter 5 the a motion planning framework for fixed-wing UAVs flying in wind is proposed. Chapter 6 describes the landing procedure of a fixed-wing UAV and how optimal landing parameters can be calculated. In Chapter 7 the implementation of the proposed framework on a real UAV platform is discussed. Chapter 8 presents experimental results which are discussed in chapter 9.

1.4 Related work

1.4.1 Motion planning

Motion planning refers to the task of finding a feasible path between a initial state and a goal for a given system. Since this is an important component of autonomous systems it has received increasing research interest lately, with an array of different algorithms and methods available.

Sampling based motion planning

Many motion planning techniques are based on discrete sampling of the continuous state - and action space. These methods are either based on random sampling, such as in Probabilistic Roadmaps [17] or Rapidly Exploring Random Trees [22] while others such as Hybrid A-star [10] use deterministic sampling.

In [16] the A-star algorithm [15] is used to find kinematically feasible trajectories for fixed-wing UAVs with a maximum turn rate while avoiding obstacles. The feasibility of the resulting path is ensured by aligning the dimensions of the sampled grid with model parameters of the given UAV.

In [35] the results in [37] are used together with A-star to generate time-optimal trajectories in the presence of wind, while also avoiding obstacles. They further define the region for where air-relative Dubins paths are not time-optimal based on the results in [25] and use these results to modify the heuristic function.

A RRT-based motion planning framework for fixed-wing UAVs, with constraints on both arrival time and final direction is proposed in [24].

Optimal control approach

The problem of finding time-optimal paths for fixed wing UAV:s in uniform winds formulated as an optimal control problem has been studied by different authors. In these works the following kinematic model is used:

$$\dot{x} = f(x,u) = \begin{bmatrix} V_a \cos\psi + W\cos\psi_w \\ V_a \sin\psi + W\sin\psi_w \\ u \end{bmatrix} \tag{1.1}$$

where the input is the turn-rate u which is constrained such as $|u| \leq \dot{\psi}_{max}$. In [25] the problem is reformulated as finding a path which intersects a virtual target moving from x_g with the same speed as the wind but in the opposite direction. It is shown that in most cases, the shortest path is a Dubin's path in the air-relative frame, but in some cases a non-Dubin's path is required to intercept the target. The optimal solution is found by defining the function

$$G(d) = T_a(d) - T_{vt}(d) \tag{1.2}$$

where d is a given distance travelled by the virtual target, $T_a(d)$ is the minimal time at which the aircraft reaches the point where the target has travelled this distance and T_{vt} is the required time for the virtual target. The optimal solution is found when $G(d) = 0$ and can be solved for using numerical root-finding techniques.

The approach in [37] is based on the observation from [34] that constant turn-rate paths in the air-relative frame correspond to *trochoidal* paths in the inertial frame. They further show that there exists an analytical solution to compute some of the optimal-path candidates, but to find all possible optimal paths a trandescendal equation has to be solved on a two dimensional grid which is computationally expensive.

1.4.2 Landing approaches

The problem of autonomously landing fixed-wing UAVs in different settings has been studied by several authors. In many of these works, the problem is defined as landing the UAV on a runway. A survey of different landing techniques is given in [13]. In [38] a framework is proposed for emergency landing of fixed-wing UAVs during thrust-lost and uniform wind. The motion planner in this work is based on the trochoidal paths discussed in [37]. The problem of landing fixed-wing UAVs on a moving ground vehicle is studied in [31].

2
Fixed-wing Unmanned Aerial Vehicles

2.1 General definitions and terminology

Fixed-wind UAVs are receiving increasing commercial and research interest, and offer a number of advantages in many use-cases. In the following sections a thorough description of general fixed-wing kinematics and control as well as a description of the specific platform used in this work will be presented. We begin with establishing some common definitions and terminology which will be used throughout this thesis. These definitions are used in many other works related to fixed-wing aircraft, such as [9], [18], [36].

2.1.1 Coordinate reference frames

Four different coordinate frames are relevant to consider for UAV applications in wind: the inertial frame which is fixed in the earth, the air-relative frame, the body frame and the wind reference frame. The body frame and wind reference frames are related through the *angle of attack* α and *sideslip* β as shown in Figure 2.1.

Definition 2.1 (Inertial frame). The inertial frame, denoted with subscript I is fixed relative to the earth. A position vector in the inertial frame is defined in the NED order as

$$\boldsymbol{p}_I = (p_N, p_E, -p_H) \tag{2.1}$$

where p_N points in the north direction, p_E points east and p_H points down towards the earth, in order to form a right hand positive coordinate system.

Definition 2.2 (Air frame). The air frame, denoted with subscript A is fixed in the air and aligned with the current direction of wind. In the case of non-zero wind, this coordinate frame moves with the same speed. This means that inertial

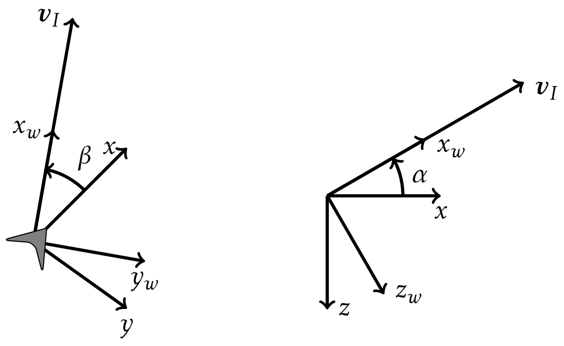


Figure 2.1: *Relation between body and wind frames*

frame coordinates become time-dependent in the air frame, and are given by

$$p_{N,A}(t) = \cos\psi_w p_{N,I} + \sin\psi_w p_{E,I} - Wt \tag{2.2}$$

$$p_{E,A}(t) = -\sin\psi_w p_{N,I} + \cos\psi_w p_{E,I} \tag{2.3}$$

where W is the wind speed and ψ_w the wind direction.

Definition 2.3 (Body frame). The body frame, denoted with subscript B is fixed in the UAV center of gravity. A position vector in the body frame is defined as

$$\boldsymbol{p}_B = (x, y, z) \tag{2.4}$$

where x points forward through the UAV, y points to the right and z points down.

Definition 2.4 (Wind reference frame). The wind reference frame, denoted with subscript W is related to the current direction of motion through the air. A position vector in the wind reference frame is defined as

$$\boldsymbol{p}_W = (x_w, y_w, z_w) \tag{2.5}$$

where x_w points in the same direction as the current velocity vector $\boldsymbol{v}_I$, y_w points to the right of x_w and z points down relative x_w and y_w.

2.1.2 Attitude representation

The attitude of the UAV is represented by the *Euler angles*.

Definition 2.5 (Euler angles). The Euler angle vector is defined as

$$\boldsymbol{\Phi} = (\phi, \theta, \psi) \tag{2.6}$$

where the *roll angle* ϕ is rotation around the north inertial axis, the *pitch angle* θ is rotation around the east inertial axis and the *yaw angle* ψ is rotation around the downwards inertial axis.

The relationship between coordinates in the body frame and inertial frame is given by the rotation matrix

$$\mathcal{R}_B^I = \mathcal{R}_\phi^x \mathcal{R}_\theta^y \mathcal{R}_\psi^z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

This attitude representation is not defined for $\theta = \pm\pi/2$. However, such attitudes were deemed very unlikely in this work as the main focus is on level flight scenarios.

2.1.3 Fixed-wing UAV

A fixed-wing UAV is equipped with two horizontal wings that are fixed in the body frame. In order to stay in the air, it needs to keep a minimum forward velocity

$$V > V_s \tag{2.8}$$

where V_s is the airframe-dependent *stall speed*. In order to navigate through the air, it is equipped with some or all of the following control surfaces:

- *Ailerons* to control ϕ
- *Elevators* to control θ
- *Rudders* to control ψ

The UAV is also equipped with one or several propellers that are used to create the thrust which increases the total energy of the system. These might be facing towards or against the direction of motion.

2.2 Trajectory following

In trajectory following, the goal is for the UAV to follow some pre-defined trajectory, which is a set of linear or curved segments in the inertial frame. This can be formulated as calculating the control signal in each time-step which minimizes the *cross-track error*

$$d(t) = \min ||\boldsymbol{p}_{I,UAV}(t) - \boldsymbol{p}_{I,traj}|| \tag{2.9}$$

where $\boldsymbol{p}_{I,traj}$ is any point on the trajectory. In this thesis, the trajectories to follow are assumed to consist of piecewise linear segments. We also chose to study the trajectory following problem only in the 2-dimensional (p_N, p_E) frame.

2.2.1 Kinematic model

When designing trajectory-following controllers for fixed-wing UAVs, the following simple kinematic model is commonly used:

$$\dot{p}_N = V_a \cos\psi + W \cos\psi_w \tag{2.10}$$

$$\dot{p}_E = V_a \sin\psi + W \sin\psi_w \tag{2.11}$$

$$\dot{\psi} = \frac{g}{V_a} \tan\phi \tag{2.12}$$

where V_a is the air-relative speed, ψ is the inertial-frame yaw angle, W is the wind magnitude, ψ_w is the wind direction and ϕ is the roll angle. The dynamic equation for the roll angle is defined as

$$\dot{\phi} = f_\phi(\phi - \phi^c) \tag{2.13}$$

where f_ϕ is defined by the inner-loop roll controller of the UAV and ϕ^c is the roll-angle command by the trajectory following controller.

2.2.2 Straight path following in wind

To study the problem of accurately following a straight path segment, it is helpful to introduce another coordinate frame which is aligned with the path segment to follow, as shown in Figure 2.2.

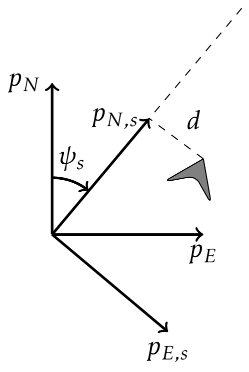


Figure 2.2: *Coordinate frame for straight path following*

The relevant equations for the control problem in this frame become

$$\dot{d} \equiv \dot{p}_{E,s} = V_a \sin(\psi - \psi_s) + W \sin(\psi_w - \psi_s) \tag{2.14}$$

$$\dot{\psi} = \frac{g}{V_a} \tan\phi \tag{2.15}$$

Assuming that $d = 0$ and $\dot{d} = 0$, we get

$$V_a \sin(\psi - \psi_s) + W \sin(\psi_w - \psi_s) = 0 \tag{2.16}$$

This means that the cross-track error is minimized when ψ converges to

$$\psi_{wca} = -\arcsin\left(\frac{W}{V_a}\sin(\psi_w - \psi_s)\right) + \psi_s \tag{2.17}$$

In the case of no wind, this simplifies to $\psi_{wca} = \psi_s$. In windy conditions however, the wind has to be compensated with a constant offset which depends on wind speed, direction and desired heading ψ_s. ψ_{wca} is called the *wind correction angle*.

2.3 ArduPlane autopilot

The ArduPlane autopilot is an open source autopilot for fixed-wing UAVs [28]. It contains high-level controllers for navigation, velocity and altitude control as well as low level logic to command the attitude and throttle of the vehicle. In the following section the underlying theory of the relevant control loops for this thesis will be presented.

2.3.1 Trajectory controller

The ArduPlane autopilot uses the L_1 control law described in [29] for trajectory following. The goal of the control loop is to follow a straight line from a start coordinate $\boldsymbol{p}_s$ to a goal coordinate $\boldsymbol{p}_g$. This is obtained by aiming towards a point P which is located at a fixed distance L_1 from the UAV. The logic behind the controller is illustrated in Figure 2.3, where $\boldsymbol{p}$ is the UAV position and ψ is the UAV yaw angle.

In the ArduPilot implementation, the distance L_1 is calculated as

$$L_1 = \begin{cases} \frac{1}{\pi}\zeta\Delta T V & \text{if} \quad |\frac{1}{\pi}\zeta\Delta T V| > |\boldsymbol{p}_g - \boldsymbol{p}| \\ |\boldsymbol{p}_g - \boldsymbol{p}| & \text{otherwise} \end{cases} \tag{2.18}$$

where $V = |\boldsymbol{v}|$, ζ is the damping factor and ΔT is the update period of the controller [29]. Wind effects are compensated by using the inertial frame velocity vector

$$\boldsymbol{v} = (V_a\cos\psi + W\cos\psi_w, V_a\sin\psi + W\sin\psi_w) \tag{2.19}$$

In each time step, the control law corresponds to following a circular segment with radius

$$R = \frac{L_1}{2\sin\eta} \tag{2.20}$$

which is tangent to $\boldsymbol{v}$ in (x, y). η is defined as the angle between the UAV velocity vector $\boldsymbol{v}$ and the line from the UAV to P. This circular segment is followed by issuing a lateral acceleration command

$$a^c = 2\frac{V^2}{L_1}\sin\eta \tag{2.21}$$

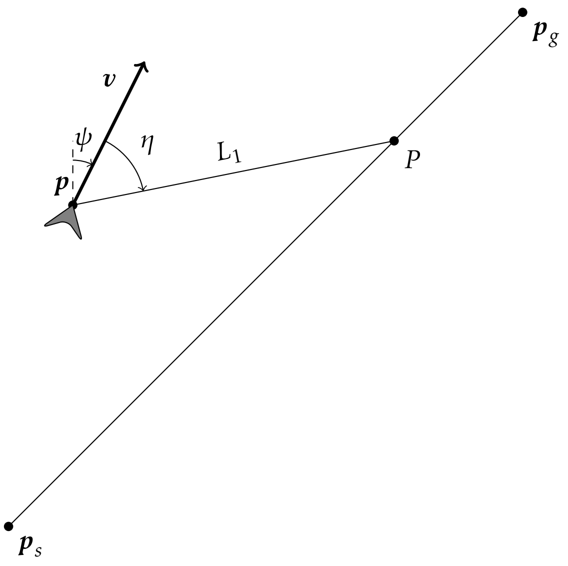


Figure 2.3: *L_1 controller logic*

The lateral acceleration command is translated to a roll command

$$\phi^c = \tan^{-1}(a^c/g) \tag{2.22}$$

where g is the gravitational constant. The low-level attitude controller is then used to track the desired roll.

In the case of a straight reference trajectory, it is shown in [30] that (2.21) can be linearized to

$$a^c \approx 2\frac{V}{L_1}\left(\dot{d} + \frac{V}{L_1}d\right) \tag{2.23}$$

which is a PD-controller. Furthermore, if inner-loop dynamics are neglected and $\boldsymbol{v}$ is assumed parallel to the reference line, $a^c \approx \ddot{d}$ and we get

$$\ddot{d} + 2\zeta\omega_n\dot{d} + \omega_n^2 d = 0 \tag{2.24}$$

with $\zeta = 1/\sqrt{2}$ and $\omega_n = \sqrt{2}V/L_1$. This is a simple second-order system where the damping is constant, and the speed depends on the ratio between V and L_1.

2.3.2 Mission representation and flight modes

A *mission* $\mathcal{M}$ is defined as

$$\mathcal{M} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_n\} \tag{2.25}$$

i. e. a sequence of *n waypoints* represented in the inertial frame as

$$\boldsymbol{p} = (p_N, p_E, -p_H, c_{wp}) \tag{2.26}$$

where c_{wp} represents the waypoint command. There are many different waypoint commands available in ArduPlane, but this work will be focused on

$$c_{wp} \in \{Waypoint, Takeoff, Land\} \tag{2.27}$$

Waypoint mode

In *waypoint* mode the trajectory following controller is used to navigate along the line from $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$. When $\boldsymbol{p}_{i+1}$ is reached, the flight mode is updated depending on the next c_{wp}. The next waypoint is assumed to be reached when

$$\|\boldsymbol{p}_{UAV} - \boldsymbol{p}_{wp}\| < R_{wp} \tag{2.28}$$

where R_{wp} is defined by the user, or passed when

$$\frac{\|\boldsymbol{p}_{UAV} \cdot \boldsymbol{p}_{wp}\|}{\|\boldsymbol{p}_{wp}\|} \geq 1 \tag{2.29}$$

where $\boldsymbol{p}_{UAV} = \boldsymbol{p} - \boldsymbol{p}_i$ and $\boldsymbol{p}_{wp} = \boldsymbol{p}_{i+1} - \boldsymbol{p}_i$.

Land mode

In *Land* mode, the plane will attempt to land at a given coordinate. The landing approach is divided into two different stages, the *approach* stage and *flare* stage.

During the approach stage, the UAV tries to accomplish the commanded *glide slope*, which is dependent on the previous waypoint position relative to the landing point. When the altitude decreases below h_{flare}, it enters the flare stage which means the throttle is completely turned off. During this stage the UAV will simply try to hold a target descent rate $\dot{h}_{flare}$ which is defined by the user [26].

3 Wind field definition and estimation

3.1 Wind field definition

The wind field is commonly defined as a time and spatially dependent vector field

$$\boldsymbol{w}(p_N, p_E, p_H, t) = \begin{bmatrix} w_N(p_N, p_E, p_H, t) \\ w_E(p_N, p_E, p_H, t) \\ w_H(p_N, p_E, p_H, t) \end{bmatrix} \tag{3.1}$$

In this thesis, the vertical component w_H will be neglected and the wind vector is written as

$$\boldsymbol{w} = W \begin{bmatrix} \cos \psi_w \\ \sin \psi_w \end{bmatrix} \tag{3.2}$$

where W is the wind magnitude and ψ_w is the wind direction. The dependency on position and time will from now on not be written out explicitly for simplicity. The wind field can be decomposed as

$$\boldsymbol{w} = \bar{\boldsymbol{w}} + \boldsymbol{w}_s \tag{3.3}$$

where $\bar{\boldsymbol{w}}$ is the mean wind field and $\boldsymbol{w}_s$ is described by some stochastic process.

3.1.1 Wind gradient

The wind magnitude W is dependent on the altitude above ground h. A simplified model of this relationship is

$$W(h) = W_0 \left(\frac{h}{h_0} \right)^a \tag{3.4}$$

where W_0 is wind measured at a reference height h_0, often 10 meters and a is the Hellman exponent. a is dependent on parameters such as the roughness of the terrain and if the location is coastal or not [7].

3.1.2 Turbulence

The stochastic components of wind is often modeled using *Dryden's Gust Model* which is stochastic process with spectral density

$$\Phi_u(\Omega) = \sigma_u^2 \frac{L_u}{\pi} \frac{1}{1 + (L_u\Omega)^2} \tag{3.5}$$

$$\Phi_w(\Omega) = \sigma_w^2 \frac{L_w}{\pi} \frac{1 + 3(L_w\Omega)^2}{(1 + (L_w\Omega)^2)^2} \tag{3.6}$$

where u is the horizontal component and w is the vertical. For altitudes below 1000 feet, the length scale of the vertical gust is $L_w = h$ and the intensity is $\sigma_w = 0.1W_{20}$ where W_{20} is the wind magnitude at 20 feet. The horizontal gust length and intensity are related to the vertical as

$$\frac{L_u}{L_w} = \frac{1}{(0.177 + 0.000823h)^{1.2}} \tag{3.7}$$

$$\frac{\sigma_u}{\sigma_w} = \frac{1}{(0.177 + 0.000823h)^{0.4}} \tag{3.8}$$

where h is measured in feet [21].

3.2 Wind estimation

Wind field estimation techniques are important for efficiently handling the effects of winds on UAVs. Fixed-wing UAVs are often equipped with a *pitot-tube* sensor which makes it possible to measure the true airspeed (TAS) of the UAV through the surrounding air.

3.2.1 Direct computation of wind field

If the inertial velocity vector $\boldsymbol{v}_I$ can be measured, e. g. with the GPS system of the UAV the wind vector can be computed directly as

$$\boldsymbol{w} = \boldsymbol{v}_I - (\mathcal{R}_B^I)^{-1}\boldsymbol{v}_B \tag{3.9}$$

where $\mathcal{R}_B^I$ is given by Equation (2.7). Assuming level flight, it is shown in [21] that the measurement error is

$$e^2 = \sigma_{\dot{p}_N}^2 + \sigma_{\dot{p}_E}^2 + \sigma_{\dot{p}_H}^2 + \sigma_{V_a}^2 + V_a^2(\sigma_\theta^2 + \sigma_\alpha^2 + \sigma_\beta^2 + \sigma_\psi^2) \tag{3.10}$$

In standard unaided GPS systems, the measurement error is approximately 0.1 m/s. Assuming the measurement error of V_a is 0.2 m/s and angles can be measured up to 1° precision, the error becomes $e^2 = 0.07 + 0.0012V_a^2$. If the airspeed is 16 m/s this corresponds to a measurement error of $e = 0.61$ m/s.

3.2.2 Estimation using Extended Kalman Filter

A more robust approach is to use an Extended Kalman Filter (EKF) to measure vehicle states. These are commonly used in autonomous systems to fuse measurements from many different sensors such as GPS, Inertial Measurement Units (IMU) and barometers. A thorough reference on the underlying theory of EKFs is given in [14].

The ArduPlane EKF implementation uses 24 different states such as attitude, velocity, position, sensor biases and wind. The different process models and measurement equations are presented in [33].

4 Motion planning theory

4.1 General definitions and terminology

In this thesis, motion planning is defined as the task of finding a path from a starting state to a goal state which fulfills a given set of constraints, while minimizing some performance measure. These constraints might include differential constraints of the system and obstacle avoidance among others. Common performance measures include minimal time or minimal energy required. We begin with introducing general definitions and terminology that are used to describe motion planning in this thesis.

4.1.1 Graph terminology

We first define the mathematical concept of graphs.

Definition 4.1 (Graph). A *graph* is defined as a set $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V}$ are the *vertices* of the graph and $\mathcal{E}$ are the *edges*. Two vertices $v_i, v_j \in \mathcal{V}$ might be connected by an edge $e_{i,j} \in \mathcal{E}$ or not connected.

Definition 4.2 (Weighted graph). In a *weighted graph*, each edge is assigned a cost $C(e) \in \mathbb{R}$.

Definition 4.3 (Directed graph). In a *directed* graph, it is possible that $c_{i,j} \neq c_{j,i}$ and there might not be an edge $e_{j,i}$ even if $e_{i,j} \in \mathcal{E}$.

Definition 4.4 (Path). A *path* in a weighted and possibly directed graph is defined as a set of edges $\mathcal{V}_p \subset \mathcal{V}$ and edges $\mathcal{E}_p \subset \mathcal{E}$ where the vertices in $\mathcal{V}_p$ are connected by the edges in $\mathcal{E}_p$. The total *cost* of a path is defined as

$$C(\mathcal{V}_p, \mathcal{E}_p) = \sum_{e \in \mathcal{E}_p} C(e) \tag{4.1}$$

4.1.2 Motion planning terminology

We further define some common terms used in motion planning.

Definition 4.5 (State and action spaces). We define the *state space* $\mathcal{X}$ and *action space* $\mathcal{U}$ as the set of obtainable states x and available actions u for the studied system. $\mathcal{X}$ can be further divided into

$$\mathcal{X} = \mathcal{X}_{free} + \mathcal{X}_{obs} \tag{4.2}$$

where $\mathcal{X}_{obs}$ are states which contain some kind of obstacle.

Definition 4.6 (Motion plan). A motion plan is defined as a sequence of states

$$\{x(t_1), \ldots, x(t_n)\} \in \mathcal{X}_{free} \tag{4.3}$$

and actions

$$\{u(t_1), \ldots, u(t_n)\} \in \mathcal{U} \tag{4.4}$$

which takes the system from a specified initial state $x(t_1) = x_i$ to a goal state $x(t_n) = x_f$ while fulfilling

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} f(x, u) dt \tag{4.5}$$

where $f(x, u)$ is the *transition function*.

4.1.3 Differential constraints

Differential constraints restrict the set of possible actions and states that the system can obtain. An important class of systems under differential constraints are *non-holonomic* systems.

Definition 4.7 (Non-holonomic system). In a *non-holonomic* system, the current state $x(t)$ is dependent on in which order the actions $u(t_i), \quad t_i < t$ where performed.

A formal definition and extensive discussion of non-holonomic systems is given in [20, Chapter 15]. Systems only capable of motion in a direction dependent on the current state, such as cars and fixed-wing UAVs belong to this class of systems.

4.2 Sampling based motion planning

Both $\mathcal{X}$ and $\mathcal{U}$ are generally continuous, and need to be sampled in some way. This means that the resulting path will only be *resolution complete*, i. e. the optimality of the plan will depend on the sampling resolution d. In sampling based motion planning a *reachability graph* is commonly used.

Definition 4.8 (Reachability graph). Given a starting state $x_0(t_0) \in \mathcal{X}_d$, we define the *reachable set* $R(x_0, \mathcal{U}_d)$ as the set of states which are reached by applying any action $u \in \mathcal{U}_d$. By incrementally calculating the reachable set for each $x \in R(x_0, \mathcal{U}_d)$ we create the *reachability tree* $\mathcal{T}_r(x_0, \mathcal{U}_d)$. The reachability tree is a directed graph where each vertex consists of a state x which is reachable from x_0 by applying some action sequence $\{u_1, \ldots, u_n\} \in \mathcal{U}_d$. By pruning any duplicate states from $\mathcal{T}_r$ we finally reach the *reachability graph* $\mathcal{G}_r(x_0, \mathcal{U}_d)$

4.2.1 Forward simulation

The next state in G_r given a specified input action u is obtained by integrating the transition function $f(x,u)$ on $[0, \Delta t]$. In practice this integral is calculated using some numerical approximation method. A common choice is the fourth-order *Runge-Kutta integration method*

$$x(\Delta t) \approx x(0) + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4) \tag{4.6}$$

where

$$\begin{aligned}
w_1 &= f(x(0), u) \\
w_2 &= f(x(0) + \frac{1}{2}\Delta t w_1, u) \\
w_3 &= f(x(0) + \frac{1}{2}\Delta t w_2, u) \\
w_3 &= f(x(0) + \Delta t w_3, u)
\end{aligned} \tag{4.7}$$

4.2.2 Motion primitives

Often it is not feasible or desirable to sample from all possible actions in $\mathcal{U}_d$. A common method is to instead create a set of *motion primitives* $\mathcal{P}$ which consists of sequences of actions that take the system from desired initial and final states.

Maneuver-based motion primitive generation is defined in [8] as the method of generating $\mathcal{P}$ based on a fixed set of maneuvers. One such maneuver is *heading change*, which is defined as taking the system from an initial heading ψ_i to a final heading ψ_f. This primitive set can be generated by solving the optimal control problem

$$\begin{aligned}
\min_{x(t),u(t),T} \quad & J = \Phi(x(T), T) + \int_0^T l(x(t), u(t))dt && && (4.8a) \\
\text{subject to} \quad & \psi(0) = 0, \quad \psi(T) = \Delta\psi && && (4.8b) \\
& \dot{x} = f(x(t), u(t)) && t \in [0, T] && (4.8c) \\
& x(t) \in \mathcal{X} && t \in [0, T] && (4.8d) \\
& u(t) \in \mathcal{U} && t \in [0, T] && (4.8e)
\end{aligned}$$

Where $\Delta\psi = \psi_f - \psi_i$ and the performance metrics $\Phi(x(T), T)$ and $l(x(t), u(t))$ are chosen such that a desired property, such as required energy or time is minimized. The motion primitive set $\mathcal{P}$ then consists of the solutions of (4.8) for different values of $\Delta\psi$.

4.3 Graph search methods

The problem of finding the minimum cost path between two vertices in a graph $\mathcal{G}$ is well studied, and there are numerous algorithms for solving it. These algorithms regularly require that $C(e) \geq 0 \forall e \in \mathcal{E}$. By using such algorithms together with the concept of reachability graphs defined in (4.8) resolution-optimal motion plans can be calculated.

4.3.1 A-star search

The *A-star* algorithm was introduced in [15] and is widely used to find the shortest path in graphs. Two important components of this algorithm is the *cost-to-come* $g(x)$ and *heuristic* $h(x)$. $g(x)$ is defined as the cost of the shortest path from the starting state x_i to x, and $h(x, x_f)$ as the cost of the shortest path from x to the goal x_f. Thus for any state x the total cost for a path through this state to the goal is given as $g(x) + h(x, x_f)$. Often the true value of $h(x, x_f)$ is not known and has to be approximated by some other function $h'(x, x_f)$. A necessary condition for optimality of the resulting path is that $h'(x, x_f)$ is *admissible*, as defined below.

Definition 4.9 (Admissible heuristic). A heuristic function $h'(x, x')$ is *admissible* if

$$h'(x, x') \leq h(x, x') \quad \forall x \tag{4.9}$$

where $h(x, x_f)$ is the true heuristic value.

An outline of motion planning with A-star is presented in Algorithm 1. The function EXPAND$(x, \mathcal{P})$ returns all states reached from x by applying motion primitives in $\mathcal{P}$ and the associated cost of each primitive. The function POP$(\mathcal{O})$ returns the state in the open set with the lowest estimated total cost, and CURRENT_COST$(x, \mathcal{O})$ returns the estimated cost currently stored for x.

4.3.2 Hybrid A-star

A disadvantage of the classical A-star formulation in motion planning is that it only allows a discrete state x_d assigned to each vertex. This was extended in [10] to the *Hybrid A-star* formulation, which instead assigns continuous states to the vertices. The difference from the classic A-star formulation is illustrated in Figure 4.1.

The Hybrid A-star algorithm also includes the concept of *analytic expansions*, which means that the model of the current system is simulated from the current state x to the goal x_f. If this simulated path is feasible, i. e. doesn't collide with obstacles the algorithm returns. It was shown that this inclusion improved

Algorithm 1 A-star based motion planning

```
Require: Motion primitive set $\mathcal{P}$, valid states $\mathcal{X}_{free}$, initial state $x_i$, final state $x_f$,
    open set $\mathcal{O}$, closed set $\mathcal{C}$
  $\mathcal{C} \leftarrow \{x_i\}$
  $\mathcal{O} \leftarrow$ EXPAND($x_s, \mathcal{P}$)
  while $\mathcal{O} \neq \emptyset$ do
      $(x, g(x)) \leftarrow$ POP($\mathcal{O}$)
      if $x == x_f$ then                                        ▷ Goal found
          return $g(x)$
      end if
      for all $(x', c') \in$ EXPAND($x, \mathcal{P}$) do
          if $x' \in \mathcal{X}_{free}$ and $x' \notin \mathcal{C}$ then
              $c_{tot} = g(x) + c' + h'(x', x_g)$                  ▷ Estimate total cost
              $c \leftarrow$ CURRENT_COST($x', \mathcal{O}$)
              if $x' \notin \mathcal{O}$ or $c > c_{tot}$ then
                  $\mathcal{O} \leftarrow \mathcal{O} \bigcup \{(x', c_{tot})\}$      ▷ Update total cost estimate
              end if
          end if
      end for
      $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{x\}$
  end while
```

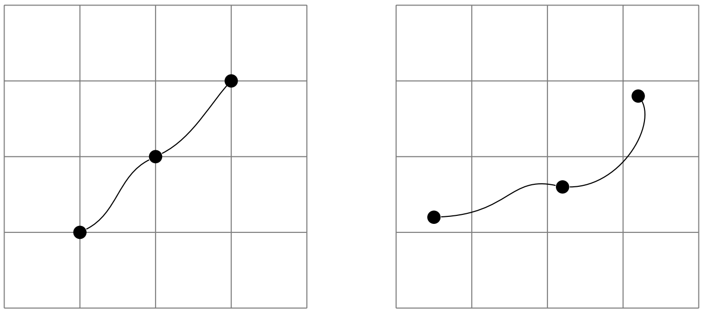

(a) Regular A-star *(b) Hybrid A-star*

Figure 4.1: Difference between regular and hybrid A-star

execution speed of the algorithm, and it also allows the goal state to be exactly reached instead of the closest discrete state.

4.3.3 Non-holonomic heuristics

A common choice of heuristic function is simply the euclidean distance $h'(x, x_f) = ||x_f - x||$. However, in many cases for non-holonomic systems this measure greatly underestimates the actual cost-to-go, which leads to unnecessary node expansions and increased computation time of the algorithm. It is therefore desirable to use another heuristic which takes the non-holonomic properties of the system into account.

Dubin's metric

The concept of Dubin's path was introduced in [11] and provides an analytical solution for the shortest path between two points (x_i, y_i, ψ_i) and (x_f, y_f, ψ_f) with a constraint on maximal turn-rate $|\dot{\psi}| \leq \dot{\psi}_{max}$. It has been widely used as a heuristic for non-holonomic systems only capable of forward motion, such as car-like robots and fixed-wing UAVs [16].

Heuristic Look-Up Table

Another efficient method for non-holonomic systems is to pre-compute the optimal cost from a number of start states to a large number of goal states and store these in a Heuristic Look-Up Table (HLUT) [19]. However, since the HLUT must be finite in size, some fallback heuristic such as euclidean distance has to be used if the value of $h(x, x')$ is not available for some x and x'. This means that some trade-off between HLUT size and algorithm efficiency has to be made, and states where the difference between $h(x, x')$ and the fallback heuristic should be prioritized for inclusion. Given a set of motion primitives $\mathcal{P}$ the HLUT can be efficiently generated using Dijkstra's algorithm. An outline of this method is given in Algorithm 2, where $\mathcal{X}$ is the set of states for which we want to generate HLUT values, and the function definitions are equal to the ones in Algorithm 1.

Algorithm 2 HLUT generation using Dijkstra's algorithm

```
Require: Motion primitive set $\mathcal{P}$, valid states $\mathcal{X}$, initial state $x_i$, open set $\mathcal{O}$, closed
  set $\mathcal{C}$
  $\mathcal{C} \leftarrow \{x_i\}$
  $\mathcal{O} \leftarrow$ EXPAND($x_i, \mathcal{P}$)
  while $\mathcal{O} \neq \emptyset$ do
      $(x, g(x)) \leftarrow$ POP($\mathcal{O}$)
      for all $(x', c') \in$ EXPAND($x, \mathcal{P}$) do
          if $x' \in \mathcal{X}$ and $x' \notin \mathcal{C}$ then
              $c_{tot} = g(x) + c'$                              ▷ Calculate cost-to-go
              $c \leftarrow$ CURRENT_COST($x', \mathcal{O}$)
              if $x' \notin \mathcal{O}$ or $c > c_{tot}$ then
                  $\mathcal{O} \leftarrow \mathcal{O} \bigcup \{(x', c_{tot})\}$          ▷ Update cost-to-go
              end if
          end if
      end for
      $\mathcal{C} \leftarrow \mathcal{C} \bigcup \{x\}$
      HLUT($x_i, x$) = $g(x)$                            ▷ Store value in HLUT
  end while
```

5 Planning in wind

5.1 State and input set definition

Based on the kinematic model in Equation (2.10) the state vector is defined as

$$x = (p_N, p_E, \psi) \tag{5.1}$$

The goal of the motion planner is to find a reference which consists of straight line-segments and takes the system from $x_i = (p_{N,i}, p_{E,i}, \psi_i)$ to $x_f = (p_{N,f}, p_{E,f}, \psi_f)$. The input u is defined as the relative positions of the start and endpoints of each line-segment. Assuming that a line-segment starts in the origin, the corresponding input is thus

$$u = (p_{N,g}, p_{E,g}) \tag{5.2}$$

Since any other path segment can be represented in this way by subtracting the start coordinates from the end coordinates this representation will be used in this chapter for ease of notation.

5.2 Closed-loop model

Based on the definition of u and the trajectory-following controller presented in Section 2.3.1 a model for the entire closed-loop system can be formulated. The desired course for the current line-segment is

$$\psi_g = \tan^{-1}(p_{E,g}/p_{N,g}) \tag{5.3}$$

By introducing a line parallel to the line-segment we can write the angle η in Figure 2.3 as

$$\eta = \eta_1 + \eta_2 \tag{5.4}$$

where η_2 is defined as the angle from the velocity vector $\boldsymbol{v}$ to this line. These angles are then given by

$$\eta_2 = \tan^{-1}\left(\frac{\cos\psi_g(V_a \sin\psi + W\sin\psi_w) - \sin\psi_g(V_a\cos\psi + W\cos\psi_w)}{\cos\psi_g(V_a\cos\psi + W\cos\psi_w) + \sin\psi_g(V_a\sin\psi + W\sin\psi_w)}\right) \tag{5.5}$$

and

$$\eta_1 = \sin^{-1}\left(\frac{p_N\sin\psi_g - p_E\cos\psi_g}{L_1}\right) \tag{5.6}$$

If roll dynamics are neglected, we then get the resulting yaw-rate by inserting the roll command from Equation (2.22) into Equation (2.10), so that

$$\dot{\psi} = \frac{a^c(x,u)}{V_a} \tag{5.7}$$

where $a^c(x,u)$ is given by Equation (2.21) with η as defined in (5.4)-(5.6). However, in a real world case there is come maximum possible yaw-rate $\dot{\psi}_{max}$ obtainable by the system, meaning that the actual yawrate will be

$$\underline{\dot{\psi}}(x,u) = \begin{cases} a^c(x,u)/V_a & |a^c(x,u)/V_a| \le \dot{\psi}_{max} \\ \operatorname{sgn}(a^c(x,u)/V_a)\dot{\psi}_{max} & \text{otherwise} \end{cases} \tag{5.8}$$

Finally, the kinematic model of the closed-loop system becomes

$$\dot{x} = f(x,u) = \begin{bmatrix} V_a\cos\psi + W\cos\psi_w \\ V_a\sin\psi + W\sin\psi_w \\ \underline{\dot{\psi}}(x,u) \end{bmatrix} \tag{5.9}$$

5.3 Motion Primitives

The motion primitive set is generated using the maneuver-base method described in Section 4.2.2. It consists of reference line-segments u which take the system from the origin to a set of desired final states, and are optimal in the sense that time is minimized while ensuring good tracking performance of the closed-loop system.

5.3.1 Wind compensation

The closed-loop kinematic model (5.9) is clearly dependent on both wind speed and direction. There are several wind-related effects that have to be taken into account when constructing the motion primitive set.

Handling different winds

During motion planning in wind, states with any possible ψ relative to the wind direction ψ_w might occur. In practice, this means that motion primitives must be generated for a set of discrete wind directions $\{\psi_{w,0}, \dots, \psi_{w,n}\}$ covering 360 degrees. The difference between discrete directions $\psi_{w,i+1} - \psi_{w,i}$ must be small enough that the closed loop system can follow a reference trajectory in wind direction $\psi_w, \quad \psi_{w,i} < \psi_w < \psi_{w,i+1}$ even though $\psi_w = \psi_{w,i}$ is used in motion primitive generation.

Since the wind speed W also affects the kinematic model, different primitive sets have to be generated for different values of W.

Course and yaw compensation

In the case of non-zero uniform wind, the required yaw angle ψ to hold a desired course ϕ_d will be different from ϕ_d as shown in Section 2.2.2. This can be compensated by formulating the initial and final constraints on desired course angle instead of yaw, where the course is given by

$$\psi_c(x) = \tan^{-1}\left(\frac{V_a \sin\psi + W \sin\psi_w}{V_a \cos\psi + W \cos\psi_w}\right) \tag{5.10}$$

This also means that the initial value of ψ should be set to the wind correction angle ψ_{wca} as defined in Equation (2.17).

5.3.2 Ensuring state connectivity

The cross-track error at the end of a line-segment can be calculated as

$$d(x_f, u) = p_{N,f} \sin\psi_g - p_{E,f} \cos\psi_g \tag{5.11}$$

If $d(x_f, u) \neq 0$ this means that $d(x_i)$ for the next line-segment will be non-zero. Since the closed-loop system is used when expanding the graph a small initial error can be handled. To ensure that the value is small enough the constraint $|d(x_f, u)| \leq d_{min}$ is added to the primitive generation optimization problem.

5.3.3 Optimal control formulation

The motion primitives are generated by solving the optimal control problem

$$\begin{aligned}
\min_{x(t),u,T} \quad & J = |d(x(T), u)|^2 + |\psi_c(x(T)) - \psi_d|^2 + \int_0^T V_a dt && && (5.12a)\\
\text{subject to} \quad & \psi(0) = \psi_{wca} && && (5.12b)\\
& \psi_c(x(0)) = 0, \quad |\psi_c(x(T)) - \psi_d| \leq \Delta\psi_{min} && && (5.12c)\\
& |d(x(T), u)| \leq d_{min} && && (5.12d)\\
& \dot{x} = f(x(t), u) && t \in [0, T] && (5.12e)\\
& x(t) \in \mathcal{X} && t \in [0, T] && (5.12f)\\
& u \in \mathcal{U} && && (5.12g)
\end{aligned}$$

for all desired values of ψ_w and desired final course ψ_d. The final term in the objective function is equal to the distance the UAV travels through the air when $t \in [0, T]$. Instead of having strict equality constraints for $d(x(T), u)$ and $\psi_c(x(T))$ they are allowed to vary a bit from the desired values to enable some tradeoff between tracking performance, final course and distance travelled.

Solving the optimal control problem

Methods commonly used to solve optimal control problems include *multiple shooting* and *direct collocation* [8]. However, there are some properties of (5.12) which makes it hard to solve which such methods. One property is that the closed-loop system is highly non-linear, especially when including the saturation from Equation (5.8). Also, normally in optimal control problems the input $u(t)$ can be chosen freely from $\mathcal{U}$ for each time-step, while in this formulation the input is forced to be a constant $u(t) = u$. This means that when transformed to a Nonlinear Program using e. g. multiple shooting, the optimization variables corresponding to $x(t)$ in each time-step all depend on the same constant u.

Derivative-free Optimization

Since all other variables in Equation (5.12) depend on the choice of u, the system can be simulated from the origin for different values of u and the objective value J calculated. Choices of u which violates the constraints can also easily be pruned. A plot of the objective function for $W = 5$, $V_a = 14$, $\psi_w = 30°$, $\psi_d = 45°$, $\Delta\psi_{min} = 10°$ and $d_{min} = 2.5$ is shown in Figure 5.1.

As can be seen the feasible region is not convex, but there is a clear global optimum. This optimization problem can be solved by using *derivative-free* optimization methods, as presented in [32]. Such methods are used to solve general optimization problems formulated as

$$\min_{x \in \mathbb{R}^n} \quad f : x \rightarrow \mathbb{R} \tag{5.13a}$$

$$\text{subject to} \quad x \in \mathcal{X}_{feasible} \tag{5.13b}$$

$$\tag{5.13c}$$

where no other information, such as the derivatives of f is available. One class of derivative-free methods, Mesh-Adaptive Direct Search (MADS) is based on creating an incrementally smaller grid around the currently optimal solution on which the objective function is evaluated. Given a good initial guess x_0 such methods are able to quickly converge to the optimal value [6].

5.4 Heuristic function

As discussed in Section 4.3.1, the choice of heuristic function is crucial in achieving good performance. The goal of the heuristic function is to estimate the length of the shortest air-relative path from an initial state x_i to a final state x_f.

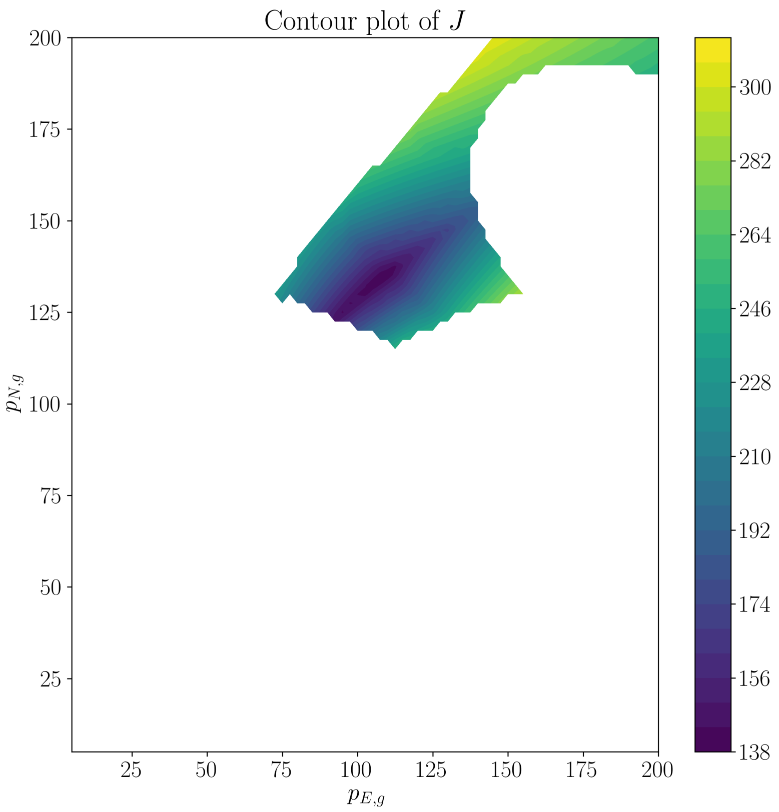


Figure 5.1: *Contour plot of optimization objective J*

5.4.1 Cost estimation for straight line-segments

As discussed in Section 2.2.2, when following a straight line-segment in wind the air-relative path travelled by the UAV will be different from the path in the inertial frame. Since the yaw angle of the UAV is equal to ψ_{wca} as defined in Equation (2.17), the velocity of the UAV along the current reference line in the inertial frame is given by

$$V_{\|} = \cos\psi_s(V_a\cos\psi_{wca} + W\cos\psi_w) + \sin\psi_s(V_a\sin\psi_{wca} + W\sin\psi_w) \tag{5.14}$$

This means that the time travelled by the UAV is equal to

$$t = \frac{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|}{V_{\|}} \tag{5.15}$$

where $\boldsymbol{p}_{i+1}$ and $\boldsymbol{p}_i$ are the end and start points of the line in inertial coordinates. Thus, the distance travelled in the air-relative frame is equal to

$$s_A = V_a t = \frac{V_a}{V_{\|}}\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\| \tag{5.16}$$

and s_A provides a good heuristic estimate for traveling along a straight path segment assuming that $\psi(x_i) = \psi(x_f) = \psi_{wca}$. This also means that the euclidean distance $\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|$ is a non-admissible heuristic if $V_a/V_{\|} < 1$, which is the case if the UAV is traveling downwind.

5.4.2 Cost estimation for arbitrary initial and final heading

Estimating the cost for traveling between states with arbitrary ψ_i and ψ_f is a more challenging problem than straight line-segments. Methods to calculate time-optimal such paths in the presence of wind are given in both [25] and [37], but since there is no analytical solution in all cases these methods rely on numerical root-finding techniques. Computing these values every time an heuristic estimate is needed was assumed too computationally expensive.

When the heuristic cannot be calculated in real-time, another option is to use a HLUT as discussed in Section 4.3.3. By using the generated motion primitives $\mathcal{P}$ when calculating costs stored in the HLUT these will be perfect estimations of the heuristic value. However, a drawback of using a HLUT is that different wind speeds W require different motion primitive sets, and thus also different HLUTs.

To estimate the cost of queries not stored in the HLUT, these queries can be projected as shown in Figure 5.2. The total heuristic value can then be estimated as

$$h'(x, x_f) = h_{HLUT}(x, x_p) + h_s(x_p, x_f) \tag{5.17}$$

where $h_s(x, x')$ is the estimated cost for a straight line-segment.

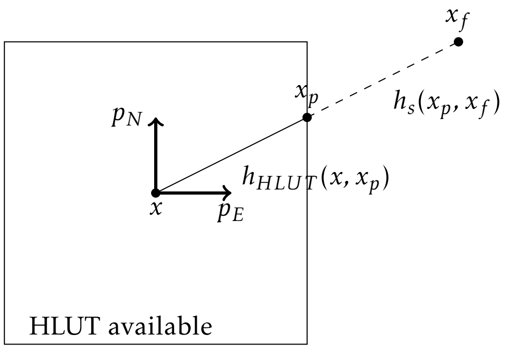


Figure 5.2: *Projection of queries on HLUT*

6 Optimal landing sequences

6.1 Problem formulation

In many previous works, the problem of landing a fixed-wing UAV on a runway is studied. However, small and light-weight UAVs such as the ones studied in this thesis can land in any area as long as the ground is flat enough. The issue is instead that there might be obstacles such as trees around the landing area which limit the possible approach directions. Wind also plays an essential role since landing downwind enables much shorter approach paths relative to the ground.

The problem of landing is thus defined as finding the optimal inputs which lands the UAV as close to the center as possible in a pre-defined landing area $\mathcal{A}$. The landing area is defined as a rectangular region with walls of height h_A, and to ensure safe landing the UAV must enter $\mathcal{A}$ above this altitude. There might also

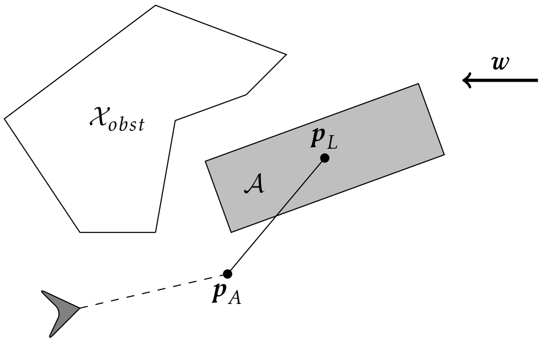


Figure 6.1: *Landing sequence definition*

be obstacle regions $\mathcal{X}_{obst}$ around the landing area where the UAV is not permitted to fly. The problem definition is illustrated in Figure 6.1.

6.2 Landing sequence

A landing sequence for fixed-wing UAVs is defined by an approach point $\boldsymbol{p}_A$ and landing point $\boldsymbol{p}_L$. These points together define an approach direction ψ_L. The landing velocity V_L depends on ψ_L, the airspeed V_a and current wind as

$$V_L = \cos\psi_L(V_a \cos\psi_{wca} + W\cos\psi_w) + \sin\psi_L(V_a \sin\psi_{wca} + W\sin\psi_w) \tag{6.1}$$

The landing sequence is divided into a approach phase and a flare phase. During the approach phase, the UAV will try to achieve the sink rate

$$\dot{h} = \frac{h_0 - h_{flare}}{\|\boldsymbol{p}_A - \boldsymbol{p}_L\| - R_{flare}} V_L \tag{6.2}$$

where h_0 is the initial altitude, h_{flare} is the flare altitude and R_{flare} is the flare distance. To ensure smooth landing, the flare phase is activated once the UAV reaches the altitude h_{flare} above the ground. In this mode it instead tries to achieve a pre-defined target descent rate

$$\dot{h} = \dot{h}_{flare} \tag{6.3}$$

which means that the flare distance is given by

$$R_{flare} = h_{flare}\frac{V_L}{\dot{h}_{flare}} \tag{6.4}$$

Due to physical limitations in the system, the landing sequence has to be defined such that

$$\dot{h} \leq \dot{h}_{max} \tag{6.5}$$

for some constant $\dot{h}_{max}$ during the initial portion before the flare.

6.3 Finding an optimal sequence

Since the goal of the landing sequence is to land as closely as possible to the center point $\boldsymbol{p}_c$ of the landing area $\mathcal{A}$, any optimal sequence will be defined by placing $\boldsymbol{p}_A$ and $\boldsymbol{p}_L$ along a line which passes through $\boldsymbol{p}_C$ and points in the direction given by ψ_L. This fact can be used to divide the problem in two parts, where first the optimal ψ_L is determined and then $\boldsymbol{p}_A$ and $\boldsymbol{p}_L$ based on the chosen direction.

6.3.1 Determining the approach direction

Any line through $\boldsymbol{p}_c$ with a given direction will cross the walls of $\mathcal{A}$ in exactly two points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, as is illustrated in Figure 6.2. We thus have the following constraints to consider:

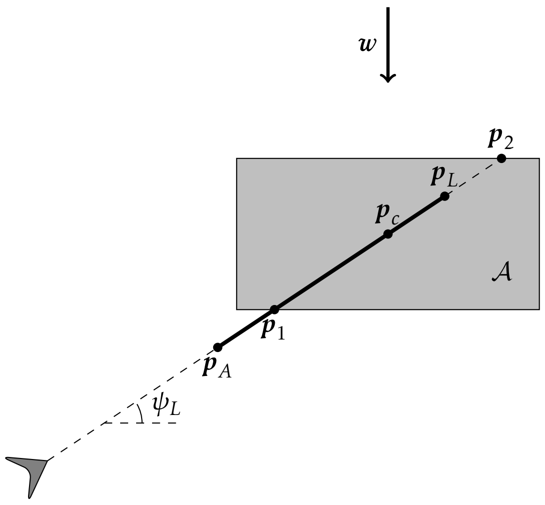


Figure 6.2: *Variables to determine optimal landing sequence*

- The distance $||\boldsymbol{p}_1 - \boldsymbol{p}_2||$ has to be large enough so that the altitude h in $\boldsymbol{p}_1$ is larger than h_A while satisfying the constraint (6.5)
- The approach direction ψ_L has to be chosen so that the initial trajectory up until $\boldsymbol{p}_A$ is not inside $\mathcal{X}_{obst}$

Assuming that the UAV is at altitude $h = h_A$, the minimum distance to the flare point is given by

$$R_{min} = (h_A - h_{flare})\frac{V_L}{\dot{h}_{max}} \tag{6.6}$$

To ensure landing in $\mathcal{A}$ it is thus required that

$$||\boldsymbol{p}_1 - \boldsymbol{p}_2|| \geq R_{min} + R_{flare} \tag{6.7}$$

where R_{flare} is given by Equation (6.4). To ensure the second constraint, a simple approach is to create lines starting in $\boldsymbol{p}_1$ with length $K(R_{min} + R_{flare})$ and direction $\psi_L + 180°$ for some $K \geq 0$ and different discrete values of ψ_L. The set of feasible approach directions $\{\psi_L\}_{feas}$ can then be found by checking each corresponding line for intersections with $\mathcal{X}_{obst}$. Finally, the optimal approach direction is chosen as

$$\psi_{L,opt} = \underset{\psi \in \{\psi_L\}_{feas}}{\arg\min}\ R(\psi) \tag{6.8}$$

where

$$R(\psi) = R_{min}(\psi) + R_{flare}(\psi) \tag{6.9}$$

6.3.2 Determining the approach points

After fixing the approach direction to $\psi_L = \psi_{L,opt}$ the next step is to calculate the optimal values of $\boldsymbol{p}_A$ and $\boldsymbol{p}_L$. Since the approach direction is fixed, the remaining variables can be redefined as

$$R_a = \|\boldsymbol{p}_A - \boldsymbol{p}_2\| \tag{6.10a}$$
$$R_l = \|\boldsymbol{p}_L - \boldsymbol{p}_2\| \tag{6.10b}$$
$$R_c = \|\boldsymbol{p}_c - \boldsymbol{p}_2\| \tag{6.10c}$$

The problem is thus finding R_a and R_l so that R_a and $|R_c - R_l|$ is minimized, while fulfilling the given constraints. The target descent rate during pre-flare is given by

$$\dot{h} = V_L \frac{h_0 - h_{flare}}{R_a - R_l - Rflare} \tag{6.11}$$

and we also require that

$$(R_a - 2R_c)\frac{\dot{h}}{V_L} \geq h_A \tag{6.12}$$

to ensure that the altitude is greater or equal than h_A in $\boldsymbol{p}_1$. The optimal values of R_a and R_l can thus be found as the solution of the optimization problem

$$\min_{R_a,R_l} \quad J = R_a^2 + \lambda|R_c - R_l|^2 \tag{6.13a}$$
$$\text{subject to} \quad R_a, R_l \geq 0 \tag{6.13b}$$
$$\dot{h} \leq \dot{h}_{max} \tag{6.13c}$$
$$(6.11)$$
$$(6.12)$$

where the weight λ is chosen to trade-off between minimizing distance to the center point and approach distance. This optimization problem has a quadratic cost function and linear constraints and as such is easy to solve using standard methods.

7 Implementation

7.1 System overview

An overview of the proposed system is shown in Figure 7.1. The UAV is assumed to be equipped with a wind estimation system which can observe the current wind vector $\boldsymbol{w}$, and a positioning system which delivers the current position of the UAV. There is also an obstacle database which contains the zones $\mathcal{X}_{obst}$ where the UAV is not allowed to fly. The user inputs a desired landing area $\mathcal{A}$ which is used to calculate an optimal landing sequence as described in Chapter 6. The resulting optimal approach point is sent as the final state to the motion planner, which calculates a plan from the current position received from the positioning system. This plan is then transformed to a waypoint mission $\mathcal{M}$ which is sent for execution to the waypoint controller.

7.2 Simulation environment

The implementation is based on the Ardupilot Software-In-The-Loop (SITL) environment [27]. This simulation environment is based on the JSBSim simulator [1], and is capable of simulating both constant and time-varying wind. The default simulation model is based on the Rascal 110 fixed-wing UAV.

7.3 Obstacle avoidance

To ensure low execution times it is crucial to use an efficient method of checking for collisions between states and $\mathcal{X}_{obst}$. In this implementation, the S2Geometry library developed by Google was used [4]. This is a C++ library which contains

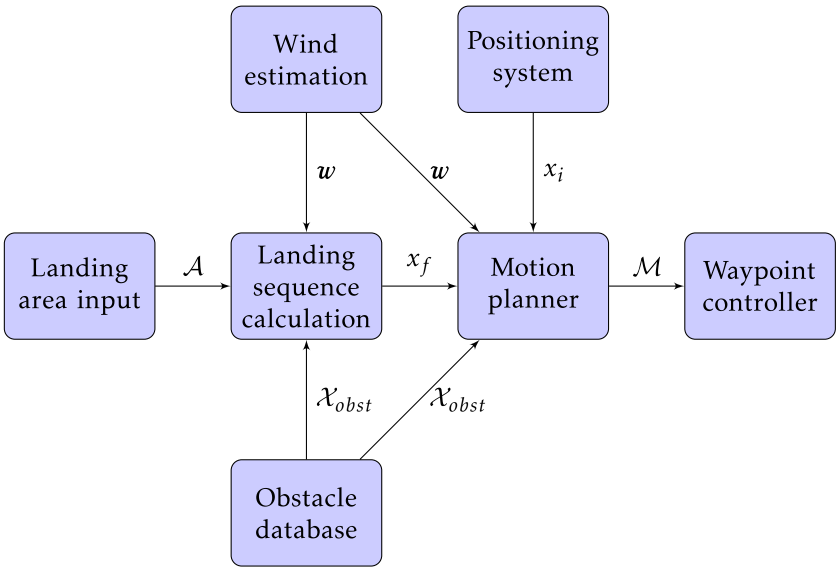


Figure 7.1: *System overview*

efficient methods to index geometrical objects of any shape, and checking for collisions between different geometries such as points, lines and polygons.

7.4 Wind estimation

In this thesis the EKF-based wind estimation system described in Section 3.2.2 was used.

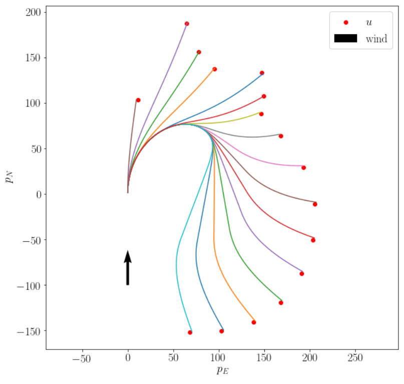
7.5 Landing sequence calculation

In order to calculate the optimal landing sequence the optimization problem (6.13) has to be solved. This problem was solved using the CasADi toolkit, which is a general toolkit for solving nonlinear optimization problems numerically [5].

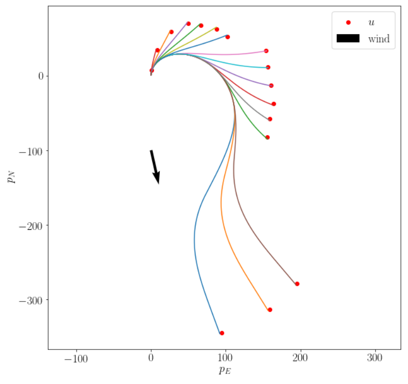
7.6 Motion planner

7.6.1 Motion Primitive Generation

Motion primitives are generated using the approach described in Section 5.3. The motion primitive set was generated for wind directions $\psi_{w,d} = \{0°, 20°, 40°, \dots 340°\}$ and desired final course $\psi_d = \{10°, 20°, \dots 160°\}$, resulting in a total of 284 primitives for a specific wind speed W. Symmetries of the system mean that motion



(a) $\psi_w = 0°$



(b) $\psi_w = 160°$

Figure 7.2: *Motion primitives for different wind directions, $W = 5\ m/s$*

primitives for $\psi_d = \{-10°, -20°, \ldots - 160°\}$ are simply found by mirroring the p_E coordinate of u. The optimization problem was solved using NOMAD [23], a C++ implementation of the MADS algorithm. The initial guess for u was defined as

$$u_0 = (d \cos \psi_d, d \sin \psi_d) \tag{7.1}$$

for the current value of ψ_d and $d = 100$ meters. Simulations of the closed-loop system for the optimal inputs u calculated for some different wind directions and $W = 5$ m/s are shown in Figure 7.2.

7.6.2 State-space discretization

To apply graph-search methods the state-space has to be discretized. In this work the values of p_N and p_E were discretized into cells of size $d = 10$ meters, and the yaw angle ψ was discretized in steps of 20°. The Hybrid A-star method was used when sampling the state space, allowing continuous values of the state vector x but assigning those to the closest discretized state.

7.6.3 State expansions

The step EXPAND($x, \mathcal{P}$) in Algorithm 1 has to take both the wind direction ψ_w and the heading ψ of x into account. Since the motion primitives in $\mathcal{P}$ are generated using initial heading $\psi_i = 0$, it is first necessary to calculate the closest relative wind direction

$$\psi_{w,rel} = \underset{\psi_{w,d} \in \{\psi_{w,d}\}}{\arg\min} |(\psi - \psi_w) - \psi_{w,d}| \tag{7.2}$$

which is used to select the motion primitives used for expansion. When mirroring primitives the wind direction also has to be mirrored so that $\psi_w' = 360° - \psi_w$. The selected motion primitives also have to be rotated so that the initial reference $u = (p_{N,g}, p_{E,g})$ is transformed to

$$u' = (\cos\psi p_{N,g} + \sin\psi p_{E,g}, -\sin\psi p_{N,g} + \cos\psi p_{E,g}) \tag{7.3}$$

Finally the expanded states and corresponding costs are found by simulating the closed-loop system (5.9) using each selected u' as input. In simulation, the actual wind direction ψ_w is used.

7.6.4 Heuristic Lookup Table

The HLUT was generated using the method in Algorithm 2. The HLUT was generated using the wind-direction $\psi_w = 0$, which means that entries have to be generated for initial values of ψ from 0° to 180° to cover all possibilities. To lookup a query $h(x, x')$ it is then necessary to rotate both x and x' by the angle ψ_w in order for the query to align with the HLUT.

To lower the amount of generated entries the discretization grid was increased to $d = 20$ meters when constructing the HLUT. The set of values for which to generate entries was selected as

$$\mathcal{X} = \{(p_N, p_E) : |p_N| \leq D \cup |p_E| \leq D\} \tag{7.4}$$

for $D = 400$ m. To ensure that HLUT are entries are available for at least states within a smaller set with $D = 200$ m, an additional A-star search was performed for each missing such state after the initial generation. For $W = 5$ m/s the resulting HLUT consists of 235359 entries. A plot of the HLUT values for initial and final heading $\psi_i = \psi_f = 0°$ can be seen in Figure 7.3. The dark values at the bottom correspond to where there are no HLUT entries available.

7.7 Waypoint controller

To send the calculated motion plan and landing sequence to the waypoint controller, these have to be converted to the MAVLink protocol which is supported by the ArduPlane autopilot [2]. This interface was implemented using the MAVROS plugin in ROS [12]. ROS is a modular framework for robotics applications, with API:s available in both Python and C++ [3].

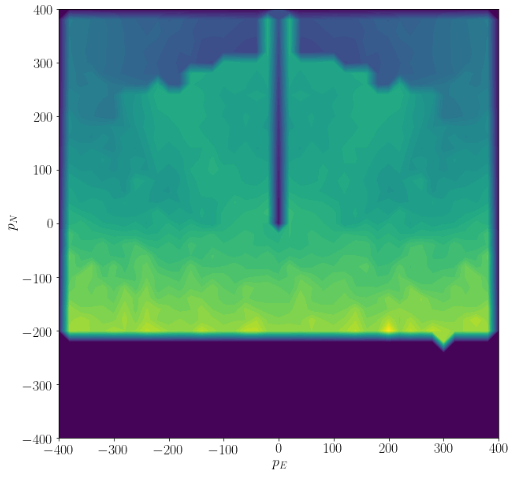


Figure 7.3: *HLUT for $W = 5$ m/s and $\psi_f = 0°$*

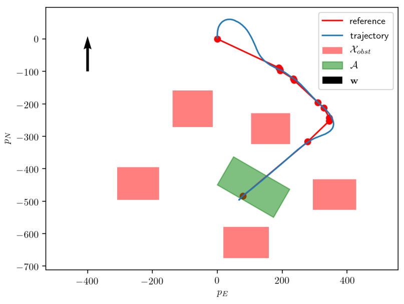
8 Results

The proposed framework was evaluated by performing a number of simulations in the ArduPlane SITL environment, in different wind conditions. Some resulting landings can be seen in Figure 8.1. The parameters used during these simulations are summarized in Table 8.1. The red dots correspond to the reference waypoints created by the motion planner, and the blue line is the actual path travelled by the UAV.

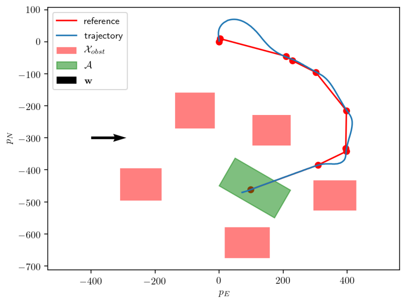
Parameter	Value	Description
x_i	$(0, 0, 0°)$	Initial state
V_a	14 m/s	Airspeed
W	5 m/s	Wind speed
h	40 m	Initial altitude
h_A	10 m	Landing area safety altitude
h_{flare}	3 m	Flare altitude
$\dot{h}_{flare}$	0.5 m/s	Flare sink-rate
$\dot{h}_{max}$	3 m/s	Maximum sink-rate
$\dot{\psi}_{max}$	17°/s	Maximum yaw-rate
λ	20	Weight factor in Equation (6.13)
$\psi_{L,d}$	10°	Approach direction discretization

Table 8.1: *Simulation parameters*

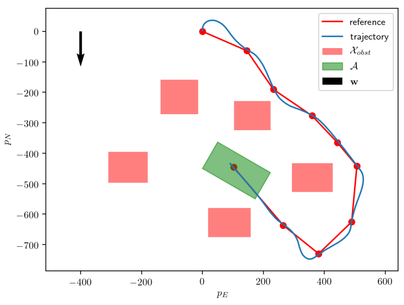
The calculated optimal landing parameters are summarized in Table 8.2. The distance $|R_a - R_b|$ corresponds to the total distance between the landing points, and the distance $|R_c - R_c|$ corresponds to the distance from the landing to the center of $\mathcal{A}$.



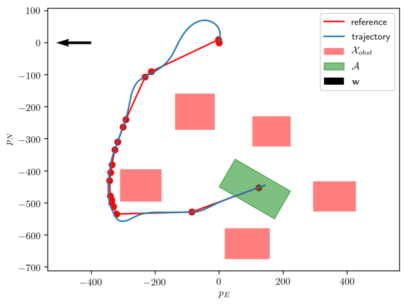
***(a)** $\psi_w = 0°$*



***(b)** $\psi_w = 90°$*



***(c)** $\psi_w = 180°$*



***(d)** $\psi_w = 270°$*

***Figure 8.1:** Resulting landing procedures for different wind directions, $W = 5\ m/s$*

ψ_w	ψ_L^*	$\lvert R_a^* - R_b^* \rvert$	$\lvert R_b^* - R_c \rvert$
0°	230°	209.8 m	42.2 m
90°	250°	173.4 m	13.5 m
180°	320°	195.5 m	14.4 m
270°	70°	173.4 m	13.5 m

***Table 8.2:** Optimal landing parameters*

9 Discussion

Bibliography

[1] Jsbsim - an open source, platform-independent, flight dynamics & control software library in c++, 2019. URL `http://jsbsim.org`.

[2] Mavlink - micro air vehicle communication protocol, 2019. URL `http://mavlink.io`.

[3] Ros - robotic operating system, 2019. URL `http://ros.org`.

[4] S2geometry - computational geometry and spatial indexing on the sphere, 2019. URL `https://s2geometry.io/`.

[5] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.

[6] Charles Audet and J. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17:188–217, 01 2006. doi: 10.1137/060671267.

[7] Francisco Bauelos-Ruedas, Cesar Angeles-Camacho, and Sebastian Rios-Marcuello. *Methodologies Used in the Extrapolation of Wind Speed Data at Different Heights and Its Impact in the Wind Energy Resource Assessment in a Region.* 06 2011. ISBN 978-953-307-483-2. doi: 10.5772/20669.

[8] Kristoffer Bergman. On motion planning using numerical optimal control, 2019. ISSN 0280-7971.

[9] Cornel-Alexandru Brezoescu. *Small lightweight aircraft navigation in the presence of wind.* PhD thesis, 2013.

[10] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *I. J. Robotic Res.*, 29:485–501, 04 2010. doi: 10.1177/0278364909359210.

[11] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957. ISSN 00029327, 10806377. URL `http://www.jstor.org/stable/2372560`.

[12] Vladimir Ermakov. Mavros - mavlink extendable communication node for ros with proxy for ground control station, 2019. URL `http://wiki.ros.org/mavros`.

[13] A. Gautam, P. B. Sujit, and S. Saripalli. A survey of autonomous landing techniques for uavs. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1210–1218, May 2014. doi: 10.1109/ICUAS.2014.6842377.

[14] F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, 2010. ISBN 9789144054896. URL `https://books.google.se/books?id=yd_2SAAACAAJ`.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

[16] M. Hwangbo, J. Kuffner, and T. Kanade. Efficient two-phase 3d motion planning for small fixed-wing uavs. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1035–1041, April 2007. doi: 10.1109/ROBOT.2007.363121.

[17] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996. doi: 10.1109/70.508439.

[18] Waqas Khan and Meyer Nahon. Dynamics modeling of a highly-maneuverable fixed-wing uav. 04 2013.

[19] R. A. Knepper and A. Kelly. High performance state lattice planning using heuristic look-up tables, Oct 2006.

[20] Steven M. La Valle. *Planning Algorithms*. 2006.

[21] Jack Langelaan, Nicholas Alley, and James Neidhoefer. Wind field estimation for small unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 34:1016–1030, 07 2011. doi: 10.2514/1.52532.

[22] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[23] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):1–15, 2011.

[24] D. Lee and D. H. Shim. Rrt-based path planning for fixed-wing uavs with arrival time and approach direction constraints. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 317–328, May 2014. doi: 10.1109/ICUAS.2014.6842270.

[25] Timothy Mcgee, Stephen Spry, and J Hedrick. Optimal path planning in a constant wind with a bounded turning rate. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, 5, 08 2005. doi: 10.2514/6.2005-6186.

[26] Ardupilot: The open source autopilot. Arduplane automatic landing, 2019. URL `http://ardupilot.org/plane/docs/automatic-landing.html`.

[27] Ardupilot: The open source autopilot. Ardupilot sitl advanced testing, 2019. URL `http://ardupilot.org/dev/docs/using-sitl-for-ardupilot-testing.html`.

[28] Ardupilot: The open source autopilot. Arduplane, 2019. URL `http://ardupilot.org/plane/index.html`.

[29] Ardupilot: The open source autopilot. Arduplane: Navigation tuning, 2019. URL `http://ardupilot.org/plane/docs/navigation-tuning.html`.

[30] Sanghyuk Park, John Deyst, and Jonathan How. *A New Nonlinear Guidance Logic for Trajectory Tracking*. doi: 10.2514/6.2004-4900. URL `https://arc.aiaa.org/doi/abs/10.2514/6.2004-4900`.

[31] Linnea Persson. Cooperative control for landing a fixed-wing unmanned aerial vehicle on a ground vehicle, 2016.

[32] Luis Miguel Rios and Nikolaos V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, Jul 2013. ISSN 1573-2916. doi: 10.1007/s10898-012-9951-y. URL `https://doi.org/10.1007/s10898-012-9951-y`.

[33] Paul Riseborough. Estimation and control library, 2019. URL `https://github.com/PX4/ecl/blob/master/EKF/documentation`.

[34] Rolf Rysdyk. Course and heading changes in significant wind. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 33:1311–1312, 07 2010. doi: 10.2514/1.48934.

[35] Martin Selecky, Petr Váňa, Milan Rollo, and Tomas Meiser. Wind corrections in flight path planning. *International Journal of Advanced Robotic Systems*, 10:1, 05 2013. doi: 10.5772/56455.

[36] Daniel Simon. *Fighter Aircraft Maneuver Limiting Using MPC - Theory and Application*. PhD thesis, 09 2017.

[37] Laszlo Techy and Craig Woolsey. Minimum-time path planning for unmanned aerial vehicles in steady uniform winds. *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, 32:1736–1746, 11 2009. doi: 10.2514/1.44580.

[38] Michael Warren, Luis Mejias, Jonathan Kok, Xilin Yang, Luis Gonzalez, and Ben Upcroft. An automated emergency landing system for fixed-wing aircraft: Planning and control. *Journal of Field Robotics*, 32:1114–1140, 12 2015. doi: 10.1002/rob.21641.