

Implementierung eines Notlandeassistenten auf Basis von Dubins-Kurven unter Berücksichtigung des Windes

Bachelorarbeit im Fach Informatik
an der FernUniversität Hagen

Felix Eckstein*

Sommer 2018

Themensteller: Prof. Dr. Wolfram Schiffmann
Lehrgebiet Rechnerarchitektur
Fachbereich Informatik

* Student im Bachelor of Science Informatik an der FernUniversität Hagen,
Matr.-#: 8161569, eckstein@embedded-engineering.de

Abstract

In this bachelor thesis an emergency landing assistant (ELA) is developed that is able to fly an aircraft in a loss of thrust emergency to a reachable emergency landing field (ELF). The path planning to reach this ELF is based on Dubins curves. The novelty concept in this thesis is the path-tracking of the autopilot component in the so called air-frame which is a reference system that moves with the wind. By using this air-frame reference, the path planning algorithm can operate with simple geometries such as circle segments and straight lines. The developed software is connected to the commercial flight simulation software X-Plane and extensive experiments are conducted to evaluate the suitability and accuracy of the air-frame based path planning and tracking in no-wind as well as in constant wind situations.

The main outcome of the experiments was, that the developed autopilot in the air-frame works very well regarding the target achievement precision. During the experiments it was also ascertained, that further research is needed to develop a model to predict the duration of descent to incorporate the wind into the path planning in advance. In the end of the thesis some proposals are made how to further improve the path planning.

INHALTSVERZEICHNIS

1	Einleitung	1
	1.1 Ziel der Arbeit	2
	1.2 Implementierung und Evaluation	3
	1.3 Aufbau der Arbeit	3
2	Grundlagen und Begriffe	4
	2.1 Grundbegriffe	4
	2.2 Geschwindigkeiten	5
	2.3 Gleitzahl, Gleitwinkel und Sinkrate	6
	2.4 Bezugssysteme	8
3	Grundlagen der Bahnplanung	11
	3.1 Dubins-Kurven	11
	3.2 Erweiterung der Dubins-Kurven in drei Dimensionen	13
	3.3 Algorithmusskizze	15
4	Software-Architektur und -Anforderungen	18
	4.1 Die Benutzerschnittstelle	19
	4.2 Komponenten der Software	20
	4.3 Anforderungen an ausgewählte Softwarekomponenten	22
5	Implementierungsdetails ausgewählter Komponenten	26
	5.1 Implementierung des Bahnplanungsmoduls	26
	5.2 Implementierung der Regelkreise	33
	5.3 Darstellung der Reglerwerte und interaktive Parametrierung	41
	5.4 Implementierung der Ablaufsteuerung	43
	5.5 Berücksichtigung des Windes und Erfassung von Messdaten	45
	5.6 Datenverbindung zum Simulator	46
	5.7 Datenhaltung und Datenausgabe	47
6	Parametrierung und Evaluation des Autopiloten	47
	6.1 Ermittlung der Grundparameter	48
	6.2 Pfadverfolgung: Abfliegen geplanter Bahnen	53
	6.3 Interpretation der Ergebnisse	59
7	Conclusio: Ergebnisse und Ausblick	64
Literatur		67
A	Anhang	A-1
	A.1 Bedienungsanleitung	A-1
	A.2 Beispiel der erfassten statistischen Daten während der Bahnverfolgung	A-2
	A.3 Datenaustausch mit X-Plane	A-7
	A.4 SW-Architektur en Détail	A-10
	A.5 Visualisierung	A-16
	A.6 Installationsanleitung und externe Bibliotheken	A-17

1 EINLEITUNG

Um ein Flugzeug stabil in der Luft zu halten wird Auftrieb benötigt, um der Schwerkraft entgegen zu wirken. Bei Motorflugzeugen erzeugt der Motor Vortrieb, der über die Tragflächengeometrie in Auftrieb gewandelt wird. Um die dabei auftretenden Reibungskräfte zu überwinden muss dem System ständig Energie in Form von Motortreibstoff zugeführt werden. Im Falle eines Motorausfalls fällt diese Energiezufuhr weg und die Auftriebs- und Vortriebsenergie muss aus anderen Quellen gedeckt werden.

Ein Motorflugzeug, dessen Antrieb durch einen Defekt ausgefallen ist, kann noch eine gewisse Strecke im Segelflug zurücklegen. Da Motorflugzeuge aufgrund ihres Designs nicht auf die Nutzung der atmosphärischen Energie ausgelegt sind, bleibt zur Überwindung der Erdanziehungskraft und des Luftwiderstandes im Wesentlichen die durch die aktuelle Höhe definierte potentielle Energie. Diese wird während des sich anschließenden (unfreiwilligen) Segelflugs permanent abgebaut und in kinetische Energie gewandelt, welche für Vortrieb und Auftrieb des antriebslosen Motorflugzeugs benutzt werden kann. Die noch zu fliegende Bahn beschreibt daher einen Sinkflug, da potentielle Lageenergie permanent umgewandelt werden muss.

Im Falle eines Notfalls mit Motorausfall sind innerhalb kurzer Zeit viele komplizierte Entscheidungen durch den Piloten zu treffen und umzusetzen:

- Identifikation des aktuellen Flugstatus und Stabilisierung
- Entscheidung für ein geeignetes und erreichbares Notlandefeld
- Bahnanplanung für den Anflug auf dieses Notlandefeld und Steuerung des Flugzeugs im Segelflug dorthin

Ein Notlandefeld ist idealerweise eine dafür ausgelegte Landebahn eines nicht zu weit entfernten Flugplatzes. Dort ist entsprechende Infrastruktur vorhanden, um das Flugzeug sicher zu landen, zu reparieren und wieder startklar zu machen. Falls eine solche Notlandebahn jedoch aufgrund von Restriktionen durch den aktuellen Flugstatus (zu geringe Flughöhe und damit zu geringe Reichweite) nicht zur Verfügung steht, so müssen geeignete Alternativen außerhalb ausgewiesener Flugplätze in Betracht gezogen werden (sog. Outfield Landing). Eine Methode zur Auswahl geeigneter Landeflächen aus Höhendaten wird z. B. in [Eckstein, Schiffmann und Wittich, 2018] vorgeschlagen.

Die Bahnanplanung zum Erreichen eines Notlandefeldes muss berücksichtigen, dass nicht nur die Koordinaten des Feldes erreicht werden können, sondern dass das Flugzeug auch in der richtigen Höhe dort eintreffen muss, um es sicher zu Boden zu bringen. Falls das Feld zu weit entfernt ist, wird das nicht angetriebene Flugzeug vor Erreichen des Ziels auf den Boden treffen, falls das Feld zu nah ist, wird das Flugzeug bei Eintreffen noch eine zu große Höhe haben, so dass es nicht an der gewünschten Position gelandet werden kann. In diesem Fall muss die Flugbahn entsprechend

verlängert werden, so dass ausreichend Höhe abgebaut wird. Eine Analyse der Erreichbarkeit von potentiellen Notlandefeldern im beschriebenen Szenario findet sich unter anderem in [Coombes, Chen und Render, 2014].

Ein weiterer wichtiger Einflussfaktor für die Bahnplanung zum gewählten Notlandefeld ist der aktuell herrschende Wind. Dieser Einfluss wird im Paper von [Coombes u. a., 2014] nicht betrachtet, sondern es wird implizit von windstillen Bedingungen ausgegangen, welche in der Praxis so fast nie vorliegen dürften.

In [Klein, Klos, Lenhardt und Schiffmann, 2018] wird ein Bahnplanungsalgorithmus skizziert, der einen möglichen Notlandeanflug für ein verunfalltes Flugzeug berechnet. Das Besondere dieses Algorithmus liegt darin, die Bahnplanung im Bezugssystem des sogenannten Air-Frames zu berechnen in dem der Windeinfluss neutralisiert ist. Der Einfluss des Windes kann durch eine Transformation der erdgebundenen Koordinaten der Zielposition in diesen Air-Frame berücksichtigt werden.

Die dort skizzierte Methode würde sich, zusammen mit einer Datenbank geeigneter Notlandefelder wie in [Eckstein u. a., 2018] skizziert, eignen einen automatischen Notlandeassistenten (Emergency Landing Assistant – ELA) zu realisieren, der im Notfall die Belastung des Piloten erheblich reduzieren könnte, um so die Sicherheit in der Luftfahrt deutlich zu erhöhen.

1.1 Ziel der Arbeit

Ziel der vorliegenden Arbeit ist es einen Autopiloten zu implementieren, der einen Flugsimulator kontrolliert. Der Autopilot fliegt im Air-Frame vorgeplante, sogenannte Dubins-Pfade mit einem simulierten Flugzeug ab. Dabei können realistische Messdaten bezüglich Genauigkeit der Flugsteuerung, Höhenverlust im Anflug und Dauer des Anflugs gewonnen werden.

Neuartig ist die Implementierung eines Autopiloten im Air-Frame: Die abzufliegende Bahn wird nicht in festen Erdkoordinaten vorgegeben, sondern relativ zur umgebenden Luft, die gegenüber der Erde vom Wind „verblasen“ wird. Dadurch lassen sich die Bahnen trotz dieser „Verblasungen“ im Wind geometrisch einfach als Abfolge von Kreissegmenten und Geradenstücken beschreiben.

Als Bahnplanungsalgorithmus wird der in [Klein u. a., 2018] skizzierte, auf Dubins-Kurven basierende Algorithmus implementiert. Es wird untersucht, wie gut die Voraussagen dieser Bahnplanung bezüglich Flugdistanz und Höhenverlust in der Praxis funktionieren. Durch Kopplung an einen Flugsimulator können in Experimenten auch Effekte der Flugphysik berücksichtigt werden, die sich einer analytischen Betrachtung in Modellrechnungen ansonsten weitgehend entziehen. Aus den im Simulator durchgeführten Messungen lassen sich Hinweise bezüglich notwendiger Korrekturen in der Bahnplanung und der Bahnverfolgung ableiten.

Eine solche Implementierung kann als Vorstufe und Basis für zukünftige automatische ELA-Systeme angesehen werden.

1.2 Implementierung und Evaluation

Es wird eine eigene Autopilot-Software erstellt die mit dem kommerziellen Flugsimulator X-Plane¹ gekoppelt wird. Damit lassen sich (simulierte) Testflüge bequem und kostengünstig durchführen, während alle relevanten Parameter inklusive des Windes kontrollierbar sind. Dieser Autopilot übernimmt im Simulator X-Plane die Ruderkontrolle eines darin simulierten Flugzeugs mit ausgefallenem Motor, um die geplante Bahn bis zum Notlandefeld abzufliegen.

In einfachen Testfällen für Geradeaus- und Kreisflug werden zunächst die grundlegenden Eigenschaften des implementierten Autopiloten und das Verhalten des Flugzeugs charakterisiert. Daraus lassen sich die Eingabeparameter für den Bahnplanungsalgorithmus bestimmen.

Als Bestandteil des Systems wird der Algorithmus aus [Klein u. a., 2018] in einem eigenen Bahnplanungsmodul implementiert. Die durch diesen Algorithmus geplanten Bahnen im Air-Frame dienen als weitere Testfälle zur Beurteilung des Autopiloten. Damit kann dessen Genauigkeit bei der Bahnverfolgung beurteilt werden.

Durch die Benutzung des Air-Frame bei der Bahnplanung und der Implementierung des Autopiloten ist zu erwarten, dass Windeffekte während des Anflugs keinen negativen Einfluss auf die Zielerreichung einer auch im Air-Frame gegebenen Zielposition haben. Es wird daher untersucht, ob sich bei Flügen mit und ohne Wind tatsächlich keine negativen Effekte für die Flugregelung ergeben. Wenn sich diese Annahme bestätigt, lässt sich über eine Voraussage der Flugdauer in Fällen mit konstantem Wind schon im Vorfeld eine entsprechende Transformation vom Earth-Frame in den Air-Frame definieren die einer Bahnkorrektur bezüglich des Windes gleichkommt. Die Flugdauerprognose selber ist jedoch nicht mehr Teil der Arbeit.

Es werden geeignete Testfälle erstellt, welche sowohl die Genauigkeit des Autopiloten als auch die Annahmen und Methoden des Bahnplanungsalgorithmus testen. Diese Testfälle werden simuliert, ausgewertet und interpretiert.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in mehrere Teile: Zunächst soll ein kurzer Abriss über die wichtigsten physikalischen Parameter und die verwendeten Begriffe gegeben werden. In einem weiteren Grundlagenkapitel werden die Eigenschaften der bei der Bahnplanung verwendeten Dubins-Kurven dargestellt und die wichtigsten Ideen des Bahnplanungsalgorithmus erläutert.

Der Hauptteil der Arbeit beschreibt, ausgehend von einem Anforderungskatalog, die Architektur und Implementierung des Autopiloten. Naturgemäß steht dabei die Realisierung der Regelungstechnik des Autopiloten zusammen mit Implementierungs-

¹ X-Plane ist eine eingetragene Marke des Entwicklers Austin Meyer dessen von ihm gegründete Firma Laminar Research die Software entwickelt und vertreibt.

details der Bahnplanung im Vordergrund. Zusätzlich wird darauf eingegangen wie das Verhältnis von Air-Frame und Earth-Frame berücksichtigt wird. Ein weiteres Kapitel des Hauptteils bildet die Beschreibung und Auswertung der durchgeführten Experimente.

Abschließend wird eine Einschätzung vorgenommen, wie die gewonnenen Erkenntnisse in eine Verbesserung des Bahnplanungsalgorithmus und in eine im Vorfeld durchzuführende Windkorrektur einfließen können.

Im Anhang werden neben einer Bedienungs- und Installationsanleitung der Software noch detailliertere Einblicke in die Softwarearchitektur gegeben.

2 GRUNDLAGEN UND BEGRIFFE

Bevor eine Software zur Flugregelung und zur Bahnplanung entwickelt werden kann, ist es notwendig einige grundsätzliche Überlegungen zu Prinzipien des Gleitflugs und zur Flugdynamik anzustellen.

Für den zu implementierenden Notlandeassistenten ist lediglich das Verhalten des Flugzeugs im Gleitflug relevant, da der ELA im vorgestellten Szenario lediglich zum Einsatz kommt, wenn der Motor ausgefallen, und daher kein Schub mehr vorhanden ist. Die für den Gleitflug notwendige Vortriebsenergie wird lediglich aus der Umwandlung potentieller in kinetische Energie gewonnen. Energiezufuhr aus thermischen oder anderen atmosphärischen Aufwinden wird ignoriert, da diese in der Praxis nicht reproduzierbar nutzbar sind.

In dieser Arbeit kann offensichtlich kein vollständiger Abriss über die relevante Flugphysik gegeben werden. Trotzdem wird versucht die relevanten Parameter wenigstens zu benennen, so dass weitere Details in der einschlägigen Fachliteratur nachgelesen werden können. Im weiteren Text werden häufig die international üblichen englischen Begriffe verwendet, da die Trefferquote bei der Recherche unter Verwendung der englischen Begriffe um ein Vielfaches höher ist, als bei Benutzung der deutschen Vokabeln.

2.1 Grundbegriffe

Um der Erdanziehung (Weight) entgegenzuwirken müssen die Tragflächen eines Flugzeugs mit Luft umströmt werden, damit sie Auftrieb (Lift) erzeugen. Das Flugzeug muss sich daher relativ zur umgebenden Luft vorwärts bewegen. Es entsteht ein „relativer Wind“, der stets senkrecht zum Auftrieb steht. Dabei muss der Luftwiderstand (Drag) überwunden werden, welcher als Kraft in Richtung relativen Windes wirkt. Die Flugzeugachse selber hat einen Winkel (Angle of Attack, AoA, α) gegenüber dem relativen Wind. Diese ist lediglich im Horizontalflug (Level Flight) identisch mit dem Flugzeugwinkel (Pitch) gegenüber der Tangentialebene auf der Erde.

Die Größe des Luftwiderstands hängt sowohl von der Geschwindigkeit relativ zur umgebenden Luft als auch vom Angle of Attack ab. Je größer der AoA, desto größer ist der Auftrieb durch die Tragflächen. Gleichzeitig nimmt jedoch auch der Drag zu. Diese rückwärtig gerichtete Kraft muss durch eine nach vorne gerichtete Kraft, den Schub (Thrust) überwunden werden.

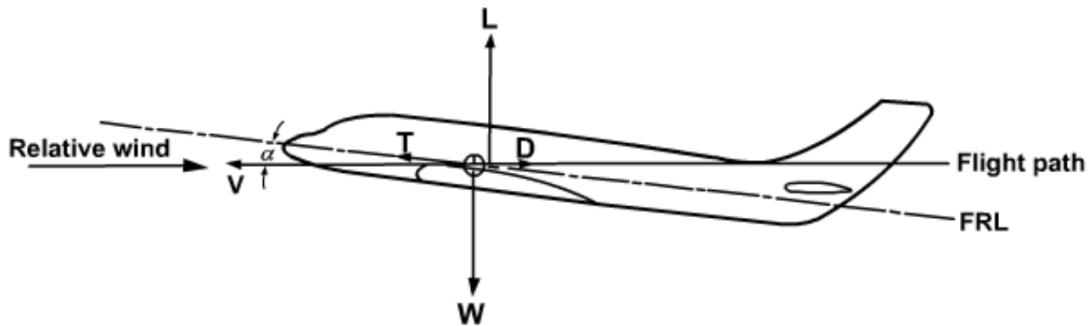


Abbildung 2.1: Kräfte auf ein Flugzeug im level flight: Lift, Weight, Drag und Thrust. Dazu die Fuselage Reference Line, der Angle of Attack α und die Velocity.
(aus [Tulapurkara, 2012, fig. 1.12])

Wie in Abbildung 2.1 dargestellt ist wirkt der Schub in Richtung der Flugzeugachse, während die Gewichtskraft senkrecht nach unten wirkt und Drag und Lift am relativen Wind bzw. senkrecht dazu ausgerichtet sind. Der Angel of Attack lässt sich über das Höhenruder (Elevator) steuern.

Auch im Falle des antriebslosen Flugzeugs muss der Drag überwunden werden, so dass noch eine ausreichende Vorwärtsbewegung stattfindet. Da kein Schub zur Verfügung steht muss zwangsläufig Höhenenergie in Geschwindigkeit gewandelt werden, so dass sich das Flugzeug im Sinkflug befindet. Es ist zu beachten, dass trotz einer gegenüber der Horizontlinie nach unten geneigten Nase immer noch ein positiver Angel of Attack α gegenüber dem, nun von unten kommenden, relativen Wind besteht, so dass auch im Sinkflug Auftrieb erzeugt wird.

2.2 Geschwindigkeiten

Bei Luft handelt es sich um ein gasförmiges Medium variabler Dichte. Die Dichte der Luft hängt von der Höhe über der Erdoberfläche und von ihrer Temperatur und Feuchtigkeit ab. Dazu kommt, dass die Luft selber sich gegenüber der Erdoberfläche bewegen kann (vulgo Wind). Die aerodynamischen Kräfte, die auf ein Flugzeug wirken sind in erster Linie von der Relativgeschwindigkeit zur umgebenden Luft und von deren Dichte abhängig.

Wesentlich für die aerodynamischen Kräfte ist die Geschwindigkeit relativ zur Luft. Leider ist die Geschwindigkeitsmessung in der Luftfahrt wesentlich komplizierter als

6 | 2.3 Gleitzahl, Gleitwinkel und Sinkrate

bei Bewegungen auf der Erdoberfläche. Es muss daher genau zwischen verschiedenen Geschwindigkeitsangaben unterschieden werden². Die für unsere Belange relevanten Geschwindigkeiten sind:

- IAS, indicated airspeed: durch Differenzdruckmessung im Flug ermittelt
- TAS, true airspeed: die Geschwindigkeit relativ zur umgebenden Luft
- GS, ground speed: die Geschwindigkeit relativ zur Erdoberfläche

Für Auftriebsberechnungen wird in der Regel die IAS herangezogen, da sich Effekte aufgrund variabler Luftdichte in unterschiedlichen Flughöhen gerade herauskürzen und so die aktuelle Flugsituation unabhängig von der aktuellen Höhe beschrieben werden kann.

Da die Navigation mit Referenz zur Erde (Earth-Frame) stattfindet und nicht relativ zur umgebenden Luft (Air-Frame) muss dazu die GS herangezogen werden. Diese GS und die TAS lassen sich bei bekannter Windgeschwindigkeit und Windrichtung leicht durch Vektoraddition ineinander überführen. Die IAS ist dagegen zur Navigation *nicht* geeignet, da sich die Umrechnung von IAS zu GS mit der aktuellen Flughöhe verändert. Als Faustformel gilt eine Abnahme der TAS (und damit bei konstantem Wind der GS) um 1,5% pro 1000 Fuß Höhenverlust. (Siehe dazu die Formelsammlung von [Williams])

In den nachfolgenden Betrachtungen muss daher immer sehr genau unterschieden werden, ob es sich bei Geschwindigkeitsangaben um die IAS für aerodynamische Berechnungen oder um die TAS/GS für die Navigation handelt.

2.3 Gleitzahl, Gleitwinkel und Sinkrate³

Die Strecke R (Range), die ein antriebsloses Flugzeug im Sinkflug zurücklegt, im Verhältnis zur auf dieser Strecke verlorenen Höhe Δh , wird als Gleitzahl E (glide ratio) bezeichnet. Aerodynamisch wird diese bestimmt durch das Verhältnis von Lift L zu Drag D und es ergibt sich $E = \frac{L}{D}$. Häufig wird statt der Gleitzahl auch deren Kehrwert, das Gleitverhältnis $\epsilon = \frac{1}{E}$ (glide number) angegeben.

Eine äquivalente Angabe ist der Gleitwinkel γ (glide angle), der definiert ist als der Winkel zwischen Horizontlinie und relativem Wind.

Aus jeder dieser Größen lässt sich bei gegebener Anfangshöhe die noch zurückzulegende Strecke ermitteln.

² Zwar nicht mit wissenschaftlichem Anspruch, dafür jedoch allgemein verständlich, werden die Zusammenhänge der einzelnen Geschwindigkeiten bei [Bislins, 2016] dargestellt.

³ Eine genaue Herleitung der Zusammenhänge dieses Abschnitts findet sich in [Yechout, 2014, ch. 3.5].

Eine weitere wichtige Angabe im Gleitflug ist die Sinkrate des Flugzeugs v_s in [$\frac{m}{s}$] (rate of descent, sink rate), die angibt, wie viel Höhe das Flugzeug pro Zeiteinheit verliert.

Unter Zuhilfenahme der Horizontalgeschwindigkeit v_h lässt sich auch die Sinkrate in die Gleitzahl überführen.

Es gilt für die genannten Größen der Zusammenhang:

$$\gamma = \arctan\left(\frac{\Delta h}{R}\right) = \arctan\left(\frac{1}{E}\right) = \arctan\left(\frac{D}{L}\right) = \arctan\left(\frac{v_s}{v_h}\right) \quad (2.1)$$

Im Verlauf dieser Arbeit hat sich der Gleitwinkel γ als gut für die Regelungstechnik zugänglich erwiesen. Daher wird meist diese Größe verwendet, obwohl das Konzept der Gleitzahl anschaulicher ist.

Für jedes Flugzeug existiert eine bestimmte Geschwindigkeit (best glide airspeed), bei der die Gleitzahl maximal ist und damit die größte Distanz aus gegebener Flughöhe zurückgelegt werden kann.

The best airspeed for gliding is one at which the airplane travels the greatest forward distance for a given loss of altitude in still air. [FAA, 2016, p. 3-20]

Diese Geschwindigkeit wird z. B. für eine Cessna 172S zu 68 KIAS (Knots Indicated Air Speed) angegeben und ist für den konstanten Sinkflug gültig ([Cessna Aircraft Company, 2004, p. 3-3]). Die optimale Geschwindigkeit sowie die zugehörige optimale Gleitzahl lassen sich aus sogenannten Gleitpolaren wie in Abbildung 2.2.A ablesen. Um die optimale Gleitzahl abzulesen, wird eine Tangente aus dem Ursprung an die Kurve angelegt. Am Berührpunkt lässt sich die optimale Gleitgeschwindigkeit und die zugehörige Sinkrate ablesen woraus sich die optimale Gleitzahl nach Gleichung (2.1) ergibt.

In Abbildung 2.2.A ist zusätzlich der Punkt geringsten Sinkens markiert, an dem die Sinkrate minimal ist. Dieser Punkt liegt immer links oberhalb vom Punkt optimalen Gleitens. Es wird in diesem Zustand zwar die Zeit in der Luft maximiert, die Reichweite des Flugzeugs ist jedoch geringer.

Neben den Gleitpolaren für den Geradeausflug sind zuweilen auch sogenannte Gleit-Hodographen verfügbar, die mehrere Gleitpolare für unterschiedliche Rollwinkel vereinen. Wie in Abbildung 2.2.B zu sehen ist, verschiebt sich die gesamte Gleitpolar im Kurvenflug nach rechts unten. Es gibt für jeden Rollwinkel eine optimale Gleitgeschwindigkeit, die größer als im Geradeausflug ist, während gleichzeitig die Gleitzahl im Kurvenflug schlechter wird.

Da es sich bei der besten Gleitgeschwindigkeit um eine aerodynamische Kenngröße handelt wird diese Geschwindigkeit in IAS angegeben. Das bedeutet, dass die TAS im optimalen Gleitflug mit der Höhe zunimmt, während gleichzeitig der Gleitwinkel konstant bleibt. Daraus ergibt sich zwingend, dass ein Flugzeug in größerer Höhe zwar schneller fliegt, dabei aber auch schneller sinkt, so dass die Reichweite des Gleitflugs

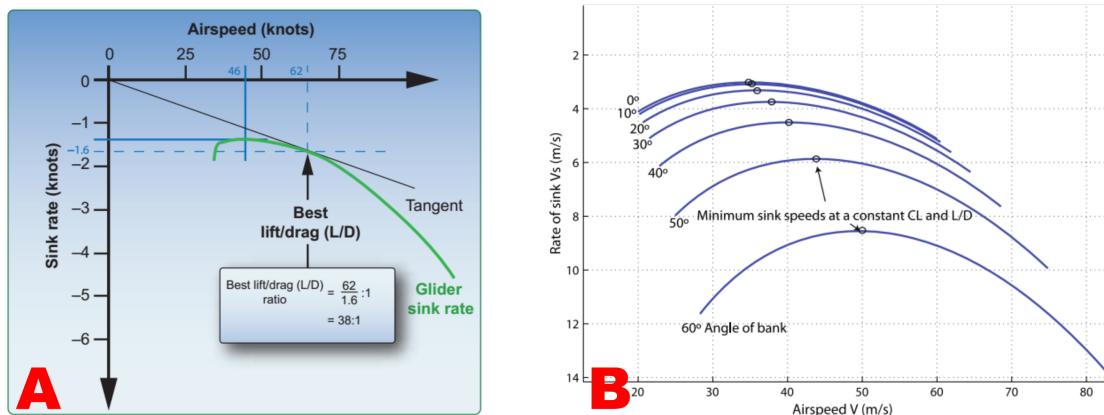


Abbildung 2.2: A: Gleitpolare für ein Segelflugzeug. Markiert sind die Punkte optimalen Gleitens und minimalen Sinkens.(aus [FAA, 2013, fig. 3-17])
 B: Hodograph für eine Cessna 182 für unterschiedliche Rollwinkel. Markiert sind die Punkte minimalen Sinkens. (aus [Coombes u. a., 2014, fig. 1])

unverändert bleibt. Die Sinkrate ist über die Höhe nicht konstant, sondern nimmt mit der Geschwindigkeit ab.⁴

Während sich die maximale Reichweite eines Flugzeugs aus gegebener Höhe einfach über den konstanten optimalen Gleitwinkel γ_{opt} voraussagen lässt, muss bei der Berechnung der Flugdauer die Änderung der Sinkrate als Funktion der Höhe $v_s = v_s(h)$ berücksichtigt werden.

2.4 Bezugssysteme

Obwohl die Erde in der alltäglichen Wahrnehmung flach erscheint, gilt es heutzutage als gesichert, dass sie eher einer Kugel gleicht.

Es ist daher üblich die Position eines Punktes auf der Erdoberfläche in einem sphärischen Koordinatensystem durch seine geographische Breite Φ (latitude) und seine geographische Länge λ (longitude) anzugeben. Das Koordinatensystem wurde durch die National Imagery and Mapping Agency im „World Geodetic System 84“ (WGS84) standardisiert. Der Ursprung des WGS84-Koordinatensystems ist der Schnittpunkt des Nullmeridians (Prime Meridian) mit dem Äquator. Die Breitenangaben liegen im Be-

4 Bei Betrachtung der Gleitpolare in Abbildung 2.2 fällt auf, dass dort die Sinkrate v_s auf der y-Achse aufgetragen ist und für die Airspeed v_h auf der x-Achse nicht angegeben ist, ob es sich um TAS oder IAS handelt. Das hat messtechnische Gründe: Die Gleitpolare wird im Windkanal ermittelt so dass sich die Höhe und damit die Luftdichte nicht ändern. Auf der x-Achse ist entsprechend die im Windkanal leicht zu erfassende TAS aufgetragen.

Genau genommen gilt eine solche Gleitpolare daher nur für eine bestimmte Luftdichte. Für Variationen über die Höhe würden sich die Punkte einer Gleitpolare entlang der Verbindungsgeraden vom Ursprung zum Punkt auf der Polare verschieben. Dadurch nehmen die absoluten Geschwindigkeiten v_h und v_s in größeren Höhen zu, während der Gleitwinkel konstant bleibt.

reich von $-90^\circ \leq \Phi \leq 90^\circ$. Positive Werte bezeichnen Punkte auf der Nordhalbkugel. Die Längenangaben liegen im Bereich von $-180^\circ \leq \lambda \leq 180^\circ$. Positive Werte bezeichnen Punkte östlich des Nullmeridians.

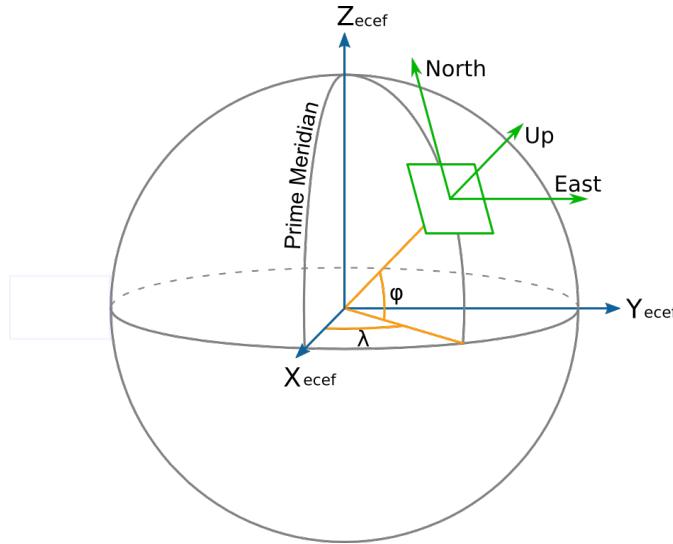


Abbildung 2.3: Verhältnis zwischen WGS84, ECEF und ENU Koordinaten. (aus [Wang u. a., 2013, fig. 3])

Dieses System wird heute in GPS-Empfängern und Kartendiensten wie Google Maps oder OpenStreetMaps zur Positionsangabe verwendet (siehe dazu z. B. [Wang u. a., 2013]). Zusätzlich wird die Höhe (altitude) in „Metern über Normal-Null“ angegeben⁵.

Äquivalent zu den Angaben im sphärischen WGS84 Koordinatensystem sind Angaben im kartesischen Earth-Centered Earth-Fixed (ECEF) System. Dieses Koordinatensystem hat den gleichen Nullpunkt wie WGS84, benutzt aber rechtwinklige kartesische Koordinaten.

Für die Navigation im Umkreis von wenigen 50 Meilen ist es ausreichend die Erde als flach zu betrachten („flat earth model“, siehe dazu [Allerton, 2009, chapter 1.4.12]). Für die Berechnungen im Zusammenhang mit dem hier zu implementierenden Notlandeassistenten ist es daher nicht notwendig die Erdkrümmung zu berücksichtigen und es kann ein einfacheres kartesisches Koordinatensystem verwendet werden. Es wird ein sogenanntes „East-North-Up“ (ENU) Koordinatensystem verwendet, dessen Koordinaten sich auf eine Tangentialebene der Erde referenzieren. Bezogen auf diese

⁵ Im vorliegenden Text wird nicht weiter auf die Problematik der Höhenangabe eingegangen. Es ist von großer Bedeutung, welche Nullebene bzw. Nullsphäre für die Höhenangabe verwendet wird, um absolute Zahlen zu vergleichen. Die verwendete Null-Sphäre unterscheidet sich dabei von Region zu Region. Für unsere Zwecke ist es ausreichend, dass konsistent eine einheitliche Nullebene verwendet wird und sich diese nicht zwischen den Berechnungen verändert.

Tangentialebene wird die Position eines Punktes mithilfe der kartesischen Komponenten East, North und Up angegeben. Dieses Bezugssystem wird als „local cartesian“ bezeichnet. Entsprechend der Konvention in der zur Umrechnung zwischen den Koordinatensystemen benutzten Bibliothek `GeographicLib` werden in der erstellten Software und im weiteren Text die kartesischen Komponenten mit x (East), y (North) und z (Up) bezeichnet. Der Zusammenhang der Koordinatensysteme wird in Abbildung 2.3 verdeutlicht.

Die Auswahl des Ursprungs der Tangentialebene für das lokale Koordinatensystem ist für die weiteren Berechnungen von entscheidender Bedeutung. Es ist zunächst nahe liegend einen festen Punkt auf der Erde festzulegen, durch den die Tangentialebene verläuft. Bei näherer Betrachtung stellt sich jedoch heraus, dass diese Festlegung für ein Flugzeug alles andere als natürlich ist.

Ein Flugzeug bewegt sich mit der umgebenden Luft. Die aerodynamischen Kräfte werden durch Bewegung der Flugzeugstruktur relativ zur Luft ausgelöst. Damit ist klar, dass ein Flugzeug zunächst keinen Bezug zu „Erdkoordinaten“ (Earth-Frame) hat, sondern seine Flugbahn besser in einem Koordinatensystem ausgedrückt werden kann, die sich zusammen mit der umgebenden Luft im Wind bewegt. Die Flugbahn lässt sich daher einfacher in „Luftkoordinaten“ (Air-Frame) beschreiben⁶.

Der Ursprungspunkt für die im Air-Frame benutzte Tangentialebene ist daher ein (virtuelles) Luftpunkt, das zu Beginn der Berechnungen an einem festen Punkt auf der Erdoberfläche im Earth-Frame übereinstimmt. Durch die vom Wind ausgelöste Luftbewegung wird dieser Ursprung des Air-Frames gegenüber dem Earth-Frame beständig bewegt. Es findet eine Translation, jedoch keine Drehung der beiden Systeme gegeneinander statt. Um von einem Koordinatensystem in das andere umzurechnen, muss diese Translation vektoriell über die Zeit auf integriert werden und kann durch einfache Vektoraddition herausgerechnet werden. Beide Systeme sind daher gleichwertig und es kann das zur Berechnung besser geeignete System frei gewählt werden.

Für die Berechnungen in dieser Arbeit ist in vielen Fällen der Air-Frame besser geeignet: Wenn ein Flugzeug beispielsweise im Air-Frame einen Kreis mit konstantem Radius fliegt, so wird dieser Kreis bei konstantem Wind im Earth-Frame zu einer Trochoide verzerrt, deren Beschreibung und Berechnung wesentlich komplizierter ist.

⁶ Das mit der umgebenden Luft bewegte Koordinatensystem wird in der einschlägigen Literatur selten verwendet und hat daher keine feste Bezeichnung. Zuweilen wird von einem „Amosphere-Fixed Reference Frame“, zuweilen vom „Wind-Frame“ gesprochen. Da der Begriff „Amosphere-Fixed Reference Frame“ sehr sperrig ist, und der Begriff „Wind-Frame“ anderweitig vorbelegt ist, soll in dieser Arbeit der Begriff „Air-Frame“ eingeführt und benutzt werden.

3 GRUNDLAGEN DER BAHNPLANUNG

Um ein Flugzeug sicher auf einer einer (Not-)Landebahn zu landen muss es zunächst dort hin fliegen. Leider sind die Bewegungen eines Flugzeugs eingeschränkt: Es kann nur gerade oder in Kurven, auf jeden Fall aber vorwärts, fliegen.

Die ausgewählte Landebahn muss so erreicht werden, dass sie in Flugrichtung des Flugzeugs ausgerichtet ist. Für die Bahnplanung ist daher neben der Start- und Endposition auch die Start- und Endrichtung entscheidend. Die Startrichtung ist durch die Startausrichtung des Flugzeugs (Heading) gegeben, die Richtung am Zielort durch die Ausrichtung der Landebahn.

Um ein Flugzeug von der aktuellen Position S mit aktueller Flugrichtung ψ_S zu einem Zielpunkt T mit Zielausrichtung ψ_T zu fliegen, muss neben der Positions- auch eine passende Richtungsänderung erfolgen. Das Flugzeug muss sich im Verlauf des Anflugs aus der aktuellen Startrichtung ψ_S in die Zielrichtung ψ_T drehen.

Eine solche Richtungsänderung kann natürlich nicht ad hoc erfolgen, sondern nur über einen Kurvenflug (Turn). Der Kurvenradius ist dabei durch die Geschwindigkeit (TAS) und den Rollwinkel (Bank, Roll, ϕ) gegeben und lässt sich über die Formel $r = \frac{V_{TAS}^2}{g \tan(\phi)}$ mit $g \approx 9,81 \frac{\text{m}}{\text{s}^2}$ berechnen (vgl. [Allerton, 2009, eq. (4.75)]). Es ist offensichtlich, dass der Kurvenradius bei gegebener Geschwindigkeit über den maximal zulässigen Rollwinkel ϕ nach unten begrenzt ist. Um sowohl die Belastung der Struktur zu begrenzen als auch den Komfort der Passagiere zu gewährleisten, wird für normale Passagierflüge der Rollwinkel im Cruise-Flight in der Regel auf $\pm 20^\circ$ begrenzt ([Allerton, 2009, p. 185]).

Die Bahnplanung für einen Landeanflug muss diese Einschränkungen in der Steuerbarkeit des Flugzeugs berücksichtigen.

3.1 Dubins-Kurven

In seinem 1957 erschienenen Artikel „On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents“ ([Dubins, 1957]) zeigt Dubins, dass für Fahrzeuge, die sich lediglich vorwärts bewegen können und einen nach unten durch r begrenzten Kurvenradius haben, die kürzeste Bahn zwischen zwei Konfigurationen immer aus der Abfolge dreier einfacher Bahnprimitive zusammengesetzt werden kann. Eine Konfiguration besteht dabei aus einer Position $[x, y]$ in der Ebene und einer Richtung ψ .

Die zu benutzenden Bahnprimitive sind:

- Straight (S): ein gerades Bahnstück
- Right (R): Ein rechtsdrehender Kreisbogen mit dem minimalem Radius r
- Left (L): Ein linksdrehender Kreisbogen mit dem minimalem Radius r

Die einzelnen Bahnsegmente gehen tangential ineinander über und die Bahnen beginnen und enden stets mit einem Kreissegment R oder L. Dazwischen wird ein weiteres Segment eingefügt, das entweder eine Tangente an den Eingangs- und den Ausgangskreis ist, oder ein gegenläufiges Mittelkreissegment. Daraus ergeben sich sechs unterschiedliche sogenannte Dubins-Pfade: RSR, LSL, RSL, LSR, RLR, LRL. Dubins zeigt, dass bei zwei beliebigen Konfigurationen in der Ebene der kürzeste Pfad immer durch eine dieser Pfadfolgen definiert wird.

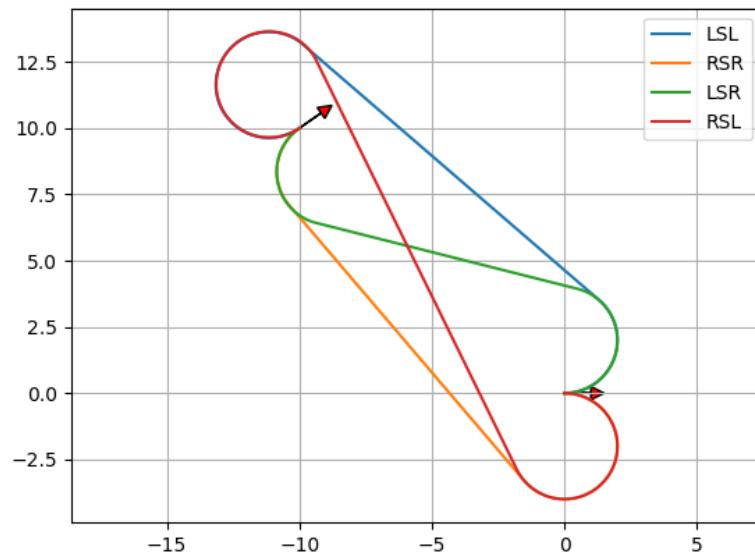


Abbildung 3.1: Beispiel der vier Dubins Pfade LSL, RSR, LSR und RSL mit geraden Anteilen.⁷

Je nach Start- und Endkonfiguration sind nicht alle sechs Pfad-Typen zulässig weil nicht alle tangentialen Übergänge zwischen Segmenten verfügbar sind. So sind die LRL- und RLR-Pfade beispielsweise nicht zulässig, wenn der minimale Abstand zwischen Eingangs- und Ausgangskreis größer als $2r$ ist. Das mittlere Kreissegment lässt sich dann nicht so einfügen, dass es sowohl den Eingangs- als auch den Ausgangskreis berührt. Die gegenläufigen Pfade RSL und LSR sind dagegen nicht zulässig, wenn sich Eingangs- und Ausgangskreis überschneiden. Dann lässt sich keine gemeinsame Tangente zwischen den Kreisen konstruieren.

Die gleichläufigen Pfade LSL und RSR dagegen sind immer zulässig: An zwei gleichläufige Kreise lässt sich unabhängig von deren Abstand immer eine gemeinsame Tan-

⁷ Bild aus <https://myenigma.hatenablog.com/entry/2017/05/01/144956>

gente konstruieren, selbst wenn sie im Grenzfall identischer Kreise eine Länge von 0 hat.

Die implementierte Bahnplanungen in dieser Arbeit beschränkt sich auf die Benutzung dieser gleichläufigen Dubins-Pfaden in der Konfiguration LSL oder RSR, selbst wenn ein anderer zulässiger Pfadtyp einen kürzeren Weg aus der Start- in die Zielkonfiguration liefern würde.

Eine ausführliche Untersuchung zulässiger Dubins-Pfade und eine Möglichkeit direkt aus Kenntnis der Anfangs- und Endkonfiguration den kürzesten Pfad-Typen auszuwählen findet sich in [Shkel und Lumelsky, 2001].

3.2 Erweiterung der Dubins-Kurven in drei Dimensionen

Zur Bahnplanung für ein Flugzeug muss das durch die Dubins-Kurven gelöste planare Problem in die dritte Dimension erweitert werden. Ein Flugzeug muss nicht nur den Zielpunkt in der richtigen Richtung erreichen, sondern dort auch in der korrekten Höhe einschweben. Im 3-dimensionalen Fall ist nicht mehr nur der kürzeste Weg von der Ausgangs- in die Zielkonfiguration gefragt, sondern es muss gleichzeitig sichergestellt werden, dass entlang dieses Weges genau die richtige Höhe abgebaut wird.

Wie in Abschnitt 2.3 dargelegt wurde, lässt sich der Höhenverlust anhand der Horizontalbewegung sehr gut voraussagen. Es existiert ein optimaler Gleitwinkel γ_{opt} der über den gesamten Sinkflug konstant bleibt. Es muss lediglich zwischen dem optimalen Gleitwinkel γ_s im Geradeausflug und dem optimalen Gleitwinkel γ_c im Kreisflug unterschieden werden.

Durch Projektion der Start- und Endposition in die Ebene und Berechnung des kürzesten Dubins Pfades kann überprüft werden, ob eine bestimmte Zielkonfiguration aus der aktuellen Flughöhe erreichbar ist. Aus den Längen der gekrümmten und des geraden Teilstücks des Dubins Pfades kann bei bekannten Gleitwinkeln γ_s und γ_c der gesamte Höhenverlust entlang des Pfades ermittelt werden. Ist der Höhenverlust kleiner oder gleich der zur Verfügung stehenden Residualhöhe, so ist die Landebahn prinzipiell erreichbar. Ist der entlang des minimalen Dubins-Pfads gegebene Höhenverlust dagegen größer als die zur Verfügung stehende Höhe, so gibt es keine Chance dieses Landefeld ohne Antrieb zu erreichen.

Durch Berechnung des kürzesten Dubins-Pfades kann somit ermittelt werden ob eine Notlandung auf einer bestimmten Piste erfolgversprechend ist, oder ob ein anderes Landefeld ausgewählt werden muss.

Für eine erfolgreiche Landung ist es nicht ausreichend lediglich genügend Reichweite zu haben, innerhalb derer das gewählte Landefeld liegt, es ist notwendig dieses Landefeld mit einer Höhe von $\sim \text{om}$ über dem Boden zu erreichen, um dort auch aufzusetzen. Falls der kürzeste Weg aus der Startkonfiguration in die Endkonfiguration nicht ausreichend viel Höhe vernichtet, muss ein „Umweg“ geflogen werden, um das

Flugzeug weiter abzusenken und in Höhe o über dem Landeplatz einzutreffen. Der kürzeste Weg, der durch den Dubins-Pfad gegeben ist muss genau soweit verlängert werden, dass auf der eingefügten Verlängerung noch genau die verbleibende Resthöhe aufgezehrt wird.

Es gibt mehrere Möglichkeiten einen Umweg zu fliegen, um Höhe zu verlieren, die sich jedoch in der Berechenbarkeit und Kontrollierbarkeit stark unterscheiden.

Die offensichtlichsten Möglichkeiten den Dubins-Pfad zu verlängern sind:

- Einfügen von vollen Kreisen am Anfang oder Ende des Dubins-Pfades. In der Projektion in die Ebene befindet sich das Flugzeug nach einem vollen Kreis in der gleichen Konfiguration wie davor, hat jedoch zwischenzeitlich eine Höhe $\Delta h = 2\pi r \tan \gamma_c$ verloren.
- Weiterfliegen in der Ursprungsrichtung, bis der Dubins-Pfad aus der inzwischen erreichten Konfiguration genau die richtige Länge hat, um die verbleibende Höhe abzubauen. Das entspricht dem Einfügen eines S-Segments am Anfang des Dubins-Pfads.
- Verschieben des Zielpunktes der Zielkonfiguration in die entgegengesetzte Landebahnrichtung soweit, dass der Dubins-Pfad zu dieser neuen Zielkonfiguration zusätzlich mit dem eingefügten geraden Bahnsegment gerade die richtige Höhe abbaut. Das entspricht dem Einfügen eines S-Segments am Ende des Dubins-Pfads und ist in Abbildung 3.2 zu sehen.

Beim Einfügen voller Kreise ist die zusätzlich abgebauten Höhen offensichtlich nicht kontinuierlich einzustellen, sondern lediglich in Schritten von $n\Delta h$, $n \in \mathbb{N}$. Dieses Methoden kann daher nur zum Einsatz kommen, wenn noch sehr viel überschüssige Höhe vorhanden ist und man vermeiden möchte sich zu weit vom gewünschten Landeplatz zu entfernen. Es ist unwahrscheinlich, dass durch das Einfügen ganzer Kreise genau die gewünschte Höhe abgebaut wird.

Bei den beiden letztgenannten Möglichkeiten lässt sich auf den ersten Blick die zusätzlich abzubauende Höhe kontinuierlich einstellen. Es ändert sich sowohl die während des Dubins-Fluges aufgezehrte Höhe als auch die Höhe, die für das eingefügte S-Segment am Anfang oder Ende benötigt wird.

Das Einfügen eines zusätzlichen Segments S_{FAL} am Ende des Anflugs ist zu präferieren, weil damit der Endanflug (Final Approach, FAL) geradeaus verläuft, und der (Auto-)Pilot so bessere Möglichkeiten hat das Flugzeug für die Landung auszurichten. Dadurch wird eine Landung direkt aus einem Kreisflug (R- oder L-Segment) heraus vermieden.

Obwohl mit dem Begriff Dubins-Kurven streng genommen nur die drei-segmentigen Kurven in der Ebene bezeichnet werden, soll der Begriff hier um die, durch die dreidimensionale Erweiterung berechneten, vier-segmentigen Bahnen LSLS und RSRS erweitert werden. Die Grundeigenschaften des tangentialen Übergangs und der Abfolge von Kreissegmenten zu Geradenstücken ist in beiden Fällen gleich.

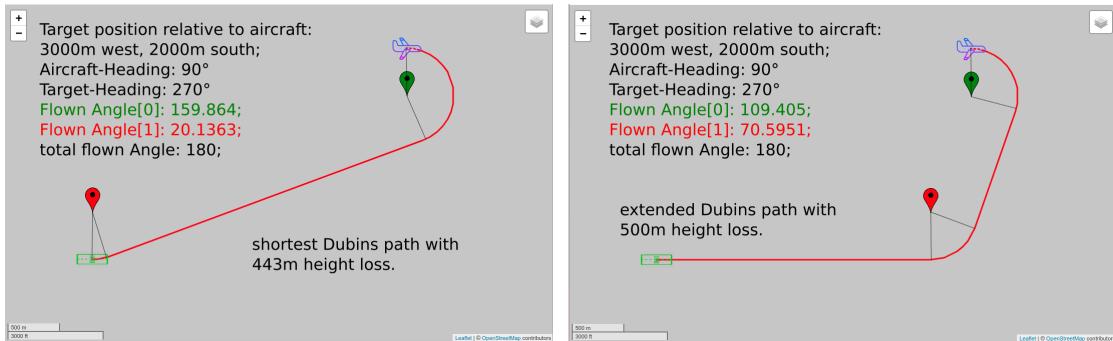


Abbildung 3.2: Vergleich des kürzesten Dubins-Pfades mit der verlängerten Bahn um 500 m Höhenverlust zu erreichen. Einfügen eines geraden Segments S_{FAL} am Ende um einen zusätzlichen Höhenverlust von 57 m zu erreichen.

3.3 Algorithmuskizze

Zur Bahnplanung mit dreidimensionalen Dubins-Kurven muss ein Algorithmus entwickelt werden, der zunächst prüft, ob das gewünschte Ziel mit der maximalen Reichweite überhaupt erreichbar ist und bei positivem Ausgang eine entsprechende LSLS- oder RSRS-Bahn berechnet, entlang derer durch Einfügen des zusätzlichen S-Segments genau die richtige Höhe abgebaut wird, um das Ziel in Höhe ~ 0 zu erreichen.

Genau das leistet der in [Klein u. a., 2018] skizzierte Algorithmus, der als Grundlage der in dieser Arbeit implementierten Bahnplanung dient. In diesem Abschnitt soll der Algorithmus lediglich skizziert werden. Genaue Berechnungen und Implementierungsdetails sind in Abschnitt 5.1 ausführlich beschrieben.

Um die Berechnungen und vor allem die Notation zu vereinfachen wird im Algorithmus zunächst eine Koordinatentransformation durchgeführt, so dass die Zielkonfiguration im Koordinatenursprung liegt und genau nach Westen (entspricht 270°) ausgerichtet ist.

Die Lage von Start- und Zielkonfiguration in der Ebene nach der Koordinatentransformation ist beispielhaft in Abbildung 3.3 zu sehen. Die Koordinaten des Zielpunkts sind $T = (0, 0)$, die des Startpunktes $S = (x_S, y_S)$.

Der Algorithmus verwendet gleichdrehende Dubins-Pfade LSL und RSR, die zu LSLS bzw. RSRS erweitert werden. Diese gleichdrehenden Pfade haben die schöne Eigenschaft, dass sich die Gesamtdrehung β des Flugzeugs entlang der Bahn nicht verändert wenn ein weiteres gerades S-Segment eingefügt wird. Wie in Abbildung 3.2 zu sehen ist, ändert sich lediglich die Aufteilung dieses Winkels zwischen dem ersten

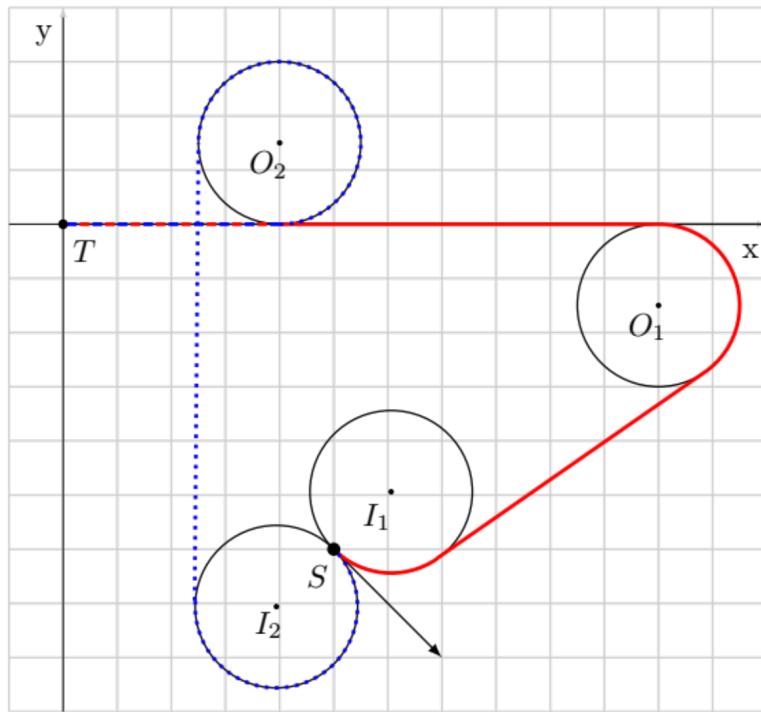


Abbildung 3.3: Lage des Zielpunkts T und des Startpunkts S nach Koordinatentransformation. Zusätzlich LSLS und RSRS Dubins-Pfade zum Anflug (aus [Klein u. a., 2018, fig. 2])

und dem zweiten Kreissegment. Die Gesamtlänge der Kreissegmente $d_C = r\beta$ und damit der Höhenverlust im Kreisflug $\Delta h_C = r\beta \tan \gamma_c$ bleiben konstant.⁸

Bei gleichdrehenden Dubins-Pfaden, entspricht die Länge des tangentialen S-Segments zwischen dem Eingangskreis I und dem Ausgangskreis O genau dem Abstand der Mittelpunkte $I = (x_I, y_I)$ und $O = (x_O, y_O)$. Daher gilt $d_T = \sqrt{(x_I - x_O)^2 + (y_I - y_O)^2} = \sqrt{\Delta x^2 + \Delta y^2}$.

Ausgehend von einem gleichdrehenden Dubins-Pfad in einer der beiden Konfigurationen LSL oder RSR wird zunächst der Gesamtdrehwinkel β und daraus die im Kreisflug aufgezehrte Höhe $\Delta h_C = r\beta \tan \gamma_c$ bestimmt. Nun kann mit dem Gleitverhältnis ϵ_s die Gesamtstrecke im Geradeausflug berechnet werden, die notwendig ist, um die verbleibende Höhe abzubauen.

Da durch die Koordinatentransformation die Endrichtung auf 270° festgelegt wurde, gilt für die y_O -Koordinate des Ausgangskreises $y_O = -r$ für einen LSLS-Anflug und

⁸ Weiter unten wird noch gezeigt werden, dass es Ausnahmen gibt, da bei der Berechnung der Gesamtdrehung eine Fallunterscheidung (5.1) zu berücksichtigen ist, welche nach Einfügen eines S-Segments anders ausfallen kann.

$y_O = r$ für einen RSRS-Anflug. Der Mittelpunkt des Eingangskreises kann leicht aus der Startposition $S = (x_S, y_S)$ und der Startrichtung ψ_S bestimmt werden.

Da auch die Richtung des einzufügenden Endegments bei 270° liegt, verändert sich durch das Einfügen lediglich die x_O -Koordinate des Ausgangskreises. Die Komponente $y_O = \mp r$ bleibt gleich. Mit Hilfe des Satzes von Pythagoras und einigen algebraischen Umformungen kann man die Länge des einzufügenden S-Segments und damit die verschobene x_O -Koordinate des Ausgangskreises berechnen, oder feststellen, dass das ausgewählte Landefeld zu weit weg ist und selbst auf dem kürzesten LSL oder RSR-Pfad nicht erreichbar ist.

Nach Abschluss der Berechnungen muss die zu Beginn durchgeführte Koordinatentransformation wieder rückgängig gemacht werden.

BERÜCKSICHTIGUNG DES WINDES Der skizzierte Algorithmus verfolgt die Idee eine Flugbahn zu planen, die auf im Air-Frame berechneten Dubins-Kurven basiert. Durch die Rechnung im Air-Frame kann der Wind bei der eigentlichen Bahnplanung ignoriert werden und muss lediglich bei der Rücktransformation in den für die Navigation wichtigen Earth-Frame berücksichtigt werden. Die Bahnberechnungen werden dadurch erheblich vereinfacht, weil ein Kurvenflug im Air-Frame einen einfachen Kreis beschreibt, während die Projektion der Bahn in den Earth-Frame bei konstantem Wind eine Trochoiden ergibt. Ein Beispiel der Bahnverzerrung im Earth-Frame unter Windeinfluss ist in Abbildung 3.4 zu gezeigt.

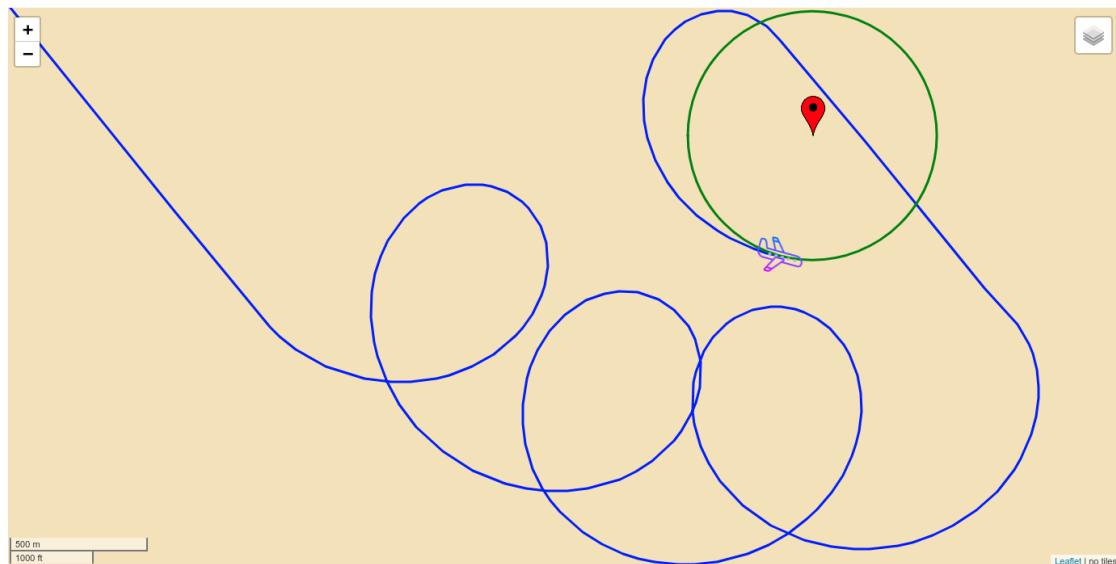


Abbildung 3.4: Beispiel für die Verzerrung von Kreisen im Air-Frame zu Trochoiden im Earth-Frame. Der grüne Kreis ist im Air-Frame geplant. Sein Mittelpunkt bewegt sich gegenüber dem Earth-Frame mit dem Wind.

Neben der einfacheren Berechnung der Bahnen bei Wind wird in [Klein u. a., 2018] auch eine einfache Möglichkeit vorgeschlagen, wie der Windeinfluss in der Bahnplanung im Vorfeld korrigiert werden kann: Wenn die Flugdauer für den Anflug bekannt ist, dann kann bei konstantem Wind die „Verblasung“ des Air-Frame gegenüber dem Earth-Frame berechnet werden. Die gewünschte Landebahn wird nun vor der Bahnplanung „gegen den Wind“ verschoben, so dass sie während des Abfliegens der Bahn wieder zurück in ihre ursprüngliche, erdfeste Position geblasen wird. Das ist möglich, da sich auch durch diese Verschiebung in der Regel keine Änderungen des Gesamt-drehwinkels im Anflug ergeben.

Der in der vorliegenden Arbeit implementierte Autopilot zur Bahnverfolgung arbeitet genau wie die hier skizzierte Bahnplanung im Air-Frame. Die durch den Wind hervorgerufenen Translation während des Anflugs wird aufintegriert und kann so im Nachhinein ausgewertet werden. Die vorgeschlagene Positionskorrektur des Zielpunkts im Air-Frame bei konstantem Wind wurde jedoch noch nicht im Bahnplanungsmodul realisiert, da die Voraussage der Flugdauer – von der diese Korrektur kritisch abhängt – bisher noch nicht zufriedenstellend gelöst werden konnte. Die vorliegende Arbeit kann hoffentlich dazu beitragen durch Messungen und Experimente ein Modell zu entwickeln, um daraus zukünftig entsprechende Korrekturen abzuleiten. Die vorausschauende Windkorrektur in der Bahnplanung liegt jedoch nicht im Fokus dieser Bachelor-Arbeit.

4 GROB-ARCHITEKTUR DER ZU ENTWICKELNDEN SOFTWARE UND ANFORDERUNGEN

Die Bahnplanung nach dem oben beschriebenen Algorithmus klingt zwar plausibel, es konnte jedoch bisher nicht verifiziert werden, ob der Ansatz für echte Notfälle mit Triebwerksausfall tatsächlich praktikabel und robust arbeiten würde. Bei positiver Evaluation des Algorithmus unter realen Einsatzbedingungen kann daraus im Weiteren ein automatischer Notlandeassistent (ELA) entwickelt werden, der durch automatisierte Landungen einen wesentlichen Beitrag zur Sicherheit in der Luftfahrt leisten kann.

Eine direkte Implementierung in ein reales Flugzeug scheidet nicht nur wegen fehlender Ressourcen (fehlendes Flugzeug) aus. Auch wäre es unverantwortlich und unzulässig einen nicht vorher in der Simulation ausführlich getesteten Algorithmus direkt in ein echtes Fluggerät zu implementieren.

Im Rahmen dieser Arbeit wurde ein System erstellt, das den Simulator X-Plane fernsteuert und sowohl die Bahnplanung als auch die Autopilot-Funktion während eines Anflugs entlang der vorberechneten Bahn übernimmt. Die Erstellung dieser Software und damit durchgeführte Experimente sind der zentrale Teil dieser Arbeit.

Im Folgenden soll der Aufbau und die SW-Architektur des Systems erklärt werden. Dabei wird weniger die einzelne Codezeile erläutert, als das High-Level Design und die Gründe für bestimmte Design-Entscheidungen. Der vollständige Source-Code für die entwickelte Software ist auf der CD zur Arbeit enthalten und auch auf Github unter <https://github.com/opt12/DubinsPilot> verfügbar.

Die Software wurde mit dem Cross-Plattform-Framework Qt 4.8 entwickelt und ist daher grundsätzlich auf Windows, OS-X und Linux Plattformen lauffähig. Getestet wurde der Code jedoch lediglich unter Linux (Ubuntu 14.04 LTS). Es ist möglich, dass gerade im Bereich der Netzwerkkommunikation auf anderen Plattformen Anpassungen notwendig werden.

4.1 Die Benutzerschnittstelle

Das System ist als GUI-Applikation ausgelegt, so dass der Benutzer interaktiv Parameter verändern kann, um deren Auswirkung zu überprüfen und so Experimente durchzuführen. Es ist auch eine Befehls-Schnittstelle über POSIX-Sockets vorbereitet, über die das Programm im Batch-Betrieb angesteuert werden könnte.

Das Programm präsentiert sich dem Benutzer als einfaches mehrseitiges Dialogfeld. In vier Dialogseiten können die Parameter der später noch beschriebenen Regler manipuliert und abgefragt werden. Auf der letzten Dialogseite können Parameter zur Bahnplanung eingetragen, Log-Dateien angelegt, und die automatische Bahnplanung und der Autopilot gestartet werden. Diese Seite bietet noch diverse weitere Steuereinstellungen, die sich zumindest in der Entwicklungs- und Experimentierphase als nützlich erwiesen haben. Ein Überblick über die Seiten des GUI ist in Abbildung 4.1 zu sehen. Eine bebilderte Bedienungsanleitung steht in Anhang A.1.

Zur Visualisierung wurde eine simple Web-App erstellt, die Positions- und Bahndaten in eine Karte einblendet.

20 | 4.2 Komponenten der Software

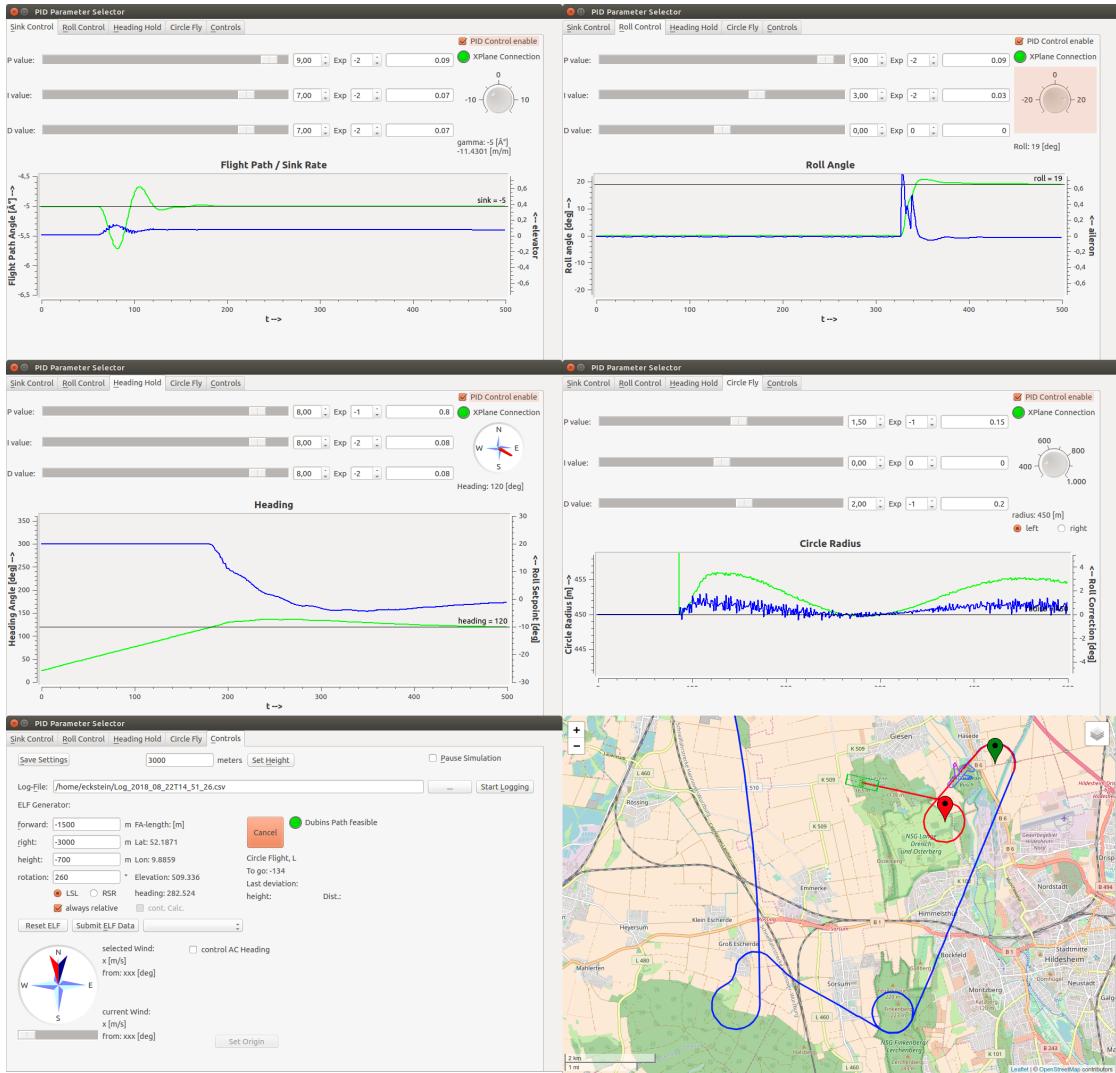


Abbildung 4.1: Überblick über die Seiten des Dialogfelds zur Steuerung des Experimentiersystems. Zusätzlich Ausschnitt aus der Visualisierung des Flugpfades per Web-App.

4.2 Komponenten der Software

Die wichtigsten Module der erstellten Software sollen kurz genannt werden. Dadurch lassen sich weiter unten formulierten Anforderungen besser sortieren und die einzelnen Klassen und Module im Source-Code lassen sich leichter zuordnen.

4.2.1 Übergreifende Komponenten

Neben den speziellen Komponenten, die in Abschnitt 5 einzeln besprochen werden, gibt es einige Programmteile die eine übergreifende Funktion erfüllen. Diese stellen eine gewisse Infrastruktur bereit, ohne Domänenlogik zu implementieren. Sie seien hier der Vollständigkeit halber kurz erwähnt:

- Dialogfeld als zentrales GUI Element
- POSIX-Socket zur Datenweitergabe an die Visualisierung; Vorbereitung zur Entgegennahme externer Befehle
- X-Plane-Kommunikationsmodul zur Datenabfrage und Fernsteuerung
- Control Automation zur Steuerung von Messungen und Logging
- Hilfsfunktionen u. a. zur Koordinatentransformation und Fehlermessung

Als zentrales Framework für die Applikationsentwicklung wird das Cross-Plattform-Framework Qt 4.8⁹ eingesetzt, da dieses eine sehr modulare Entwicklung mit einer losen Kopplung aller Komponenten ermöglicht. Auf weitgehende Modularität der einzelnen Bestandteile wurde von Anfang an großer Wert gelegt, da so sichergestellt ist, dass der entwickelte Code für Bahnplanung und Flugregelung leicht erweitert werden oder auch z. B. in eine embedded Applikation für ein reales Fluggerät überführt werden kann.

4.2.2 Das Autopilot Modul

Das zentrale Modul der Applikation ist der Autopilot, der ein Flugzeug automatisch entlang einer im Air-Frame vorberechneten Bahn fliegt. Dieses Modul besteht selber wiederum aus zwei Hauptkomponenten. Diese werden in Kapitel 5 ausführlich erklärt.

- (kaskadierte) PID Regler für die Flug- und Bahnregelung
 - Sink Control zur Regelung des Gleitwinkels
 - Roll Controll zur Regelung des Rollwinkels zur Richtungssteuerung
 - Heading Hold zur Regelung eines Geradeausflugs in eine bestimmte Richtung
 - Localizer Interception zum Anflug eines Punktes in einer vorgegebenen Richtung
 - Circle Flight zum Kreisflugs um einen Mittelpunkt mit vorgegebenem Radius
- Flugablaufsteuerung zum automatischen Abfliegen der Dubins-Kurven

⁹ <https://doc.qt.io/archives/qt-4.8/>

4.2.3 Das Bahnplanungsmodul

Zur Bahnplanung eines möglichen Anflugs wird ein Bahnplanungsmodul erstellt. In diesem ist der in Abschnitt 3.3 skizzierte Algorithmus nach [Klein u. a., 2018] umgesetzt, so dass er zusammen mit dem Autopiloten evaluiert werden kann.

Das Bahnplanungsmodul berechnet auf Grundlage der aktuellen Flugzeugposition, der Zielposition und diversen Bahnparametern einen verlängerten Dubins-Pfad als Anflug. Die Bahnparameter sind dabei der zu verwendende Kurvenradius und der im Geradeaus- und im Kreisflug zu verwendende Gleitwinkel. Aufgrund dieser Parameter und der zu überwindenden Höhendifferenz wird die vom Autopiloten zu verfolgende Flugbahn geplant.

4.2.4 Visualisierung als entkoppelte Web-App

Zur Visualisierung der geplanten und der tatsächlichen Flugbahn wurde eine einfache Web-App entwickelt. Sie besteht aus den beiden Komponenten:

- React-Web-App zur Darstellung der Flugbahnen im Browser
- Express-Proxy Server zur Bereitstellung der Daten

Die Entscheidung die Visualisierung als Web-App auszuführen wurde getroffen, weil zur Geodatenvisualisierung die einfach zu benutzende Leaflet-Bibliothek existiert, mit der in früheren Projekten schon sehr gute Erfahrungen gesammelt wurden. Das Gleiche gilt für den Einsatz eines Express-Servers als Proxy zwischen der Datenausgabe der Autopilot-App per POSIX-Socket und der Bereitstellung der Daten über eine REST-API über HTTP an die Web-App.

Beide Komponenten, insbesondere der REST-Server ließen sich auch direkt in die Autopilot-App integrieren. Der dafür notwendige Aufwand übersteigt aber den zusätzlichen Nutzen im vorliegenden Anwendungsszenario bei Weitem.

4.3 Anforderungen an ausgewählte Softwarekomponenten

Zunächst muss geklärt werden, welche Anforderungen die zu entwickelnde Applikation erfüllen muss, um den oben erläuterten Algorithmus zur Bahnplanung zu evaluieren und in Flugversuchen danach berechnete Anflüge durchzuführen. Die zentralen Anforderungen werden zunächst aufgezählt, bevor die Konzepte erläutert werden, mit denen sie erfüllt werden.

4.3.1 Anforderungen an den Autopiloten zum Abfliegen von Dubins-Kurven

Der Bahnplanungsalgorithmus basiert auf einer genauen Voraussage der zurückgelegten Strecke bei definiertem Höhenverlust. Dazu ist es notwendig den Gleitwinkel

γ auf einen definierten Wert zu regeln. Wenn die entsprechenden Hodographen für den eingesetzten Flugzeugtyp vorliegen, so kann die Regelung direkt auf „best glide“ eingestellt werden. Falls die Hodographen nicht vorliegen, können sie durch entsprechende Experimente im Flugversuch ermittelt werden.

Anforderung 1: Einstellen eines definierten Gleitwinkels Der Autopilot soll den Gleitwinkel während des Geradeausflugs auf einen definierten Wert γ_S und während des Kurvenflugs auf einen definierten Wert γ_C regeln. Als Steuerelement soll das Höhenruder (Elevator) dienen.

Der Kurvenflug des Flugzeugs wird über den Rollwinkel geregelt. Dieser wird allein durch Kontrolle des Querruders (Aileron) gesteuert. Es wird darauf verzichtet sogenannte „coordinated turns“ zu fliegen, bei denen auch das Seitenruder eingesetzt wird.

Anforderung 2: Einstellen eines definierten Rollwinkels Der Autopilot soll einen bestimmten Rollwinkel ausregeln können. Als Steuerelement soll das Querruder (Aileron) dienen.

Die Grundelemente der berechneten Bahnen sind gerade Strecken und Kreissegmente. Diese geometrische Form der Elemente bezieht sich dabei auf den Air-Frame während sie bei Bezug auf den Earth-Frame unter Windeinfluss verzerrt werden.

Anforderung 3: Heading-Hold im Air-Frame Der Autopilot soll die Flugrichtung so regeln können, dass im Air-Frame eine gerade Strecke geflogen wird.

Anforderung 4: Circle-Flight im Air-Frame Der Autopilot soll die Flugrichtung so regeln können, dass im Air-Frame ein Kreissegment mit vorgegebenem Radius geflogen wird. Die Drehrichtung soll einstellbar entweder L oder R sein.

Anforderung 5: Berücksichtigung der Scherung der Geometrie bei Projektion in den Earth-Frame Der Autopilot soll den Einfluss des Windes berücksichtigen wenn Koordinatentransformationen vom Wind- in den Earth-Frame und umgekehrt erfolgen.

Die durch den Algorithmus geplanten Bahnen bestehen immer aus vier Bahnprimitiven in der Reihenfolge LSLS bzw. RSRS. Jedes dieser Bahnprimitive kann eine beliebige Länge bzw. einen beliebigen Drehwinkel haben. Diese Werte können bis auf 0 schrumpfen. Die Kreisflüge können auch einzeln mehr als 360° Drehwinkel haben, wenn bei der Bahnplanung eine entsprechende Korrektur notwendig wurde. Der Autopilot muss in der Lage sein diese Bahnprimitive abzufliegen und die Sollwerte der untergeordneten Regler entsprechend einzustellen. Um eine geplante Bahn aus der aktuellen Flugsituation heraus abfliegen zu können ist darüber hinaus eine Ablaufsteuerung erforderlich, die eine Sequenz von Bahnprimitiven nacheinander als Vorgabe auswählt und am Ende eines Primitivs automatisch auf das nächste umschaltet. Bei einer solchen Hintereinanderschaltung von Bahnprimitiven kann es vorkommen,

dass der Endpunkt eines Bahnprimitivs nicht genau erreicht wird. Dies kann durch Ungenauigkeiten im Regelkreis, durch Überschwinger oder durch andere externe Störungen passieren. Es ist ein toleranter Mechanismus zu implementieren, der feststellt, dass ein gewünschter Punkt im aktuellen Flugmodus (geradeaus- oder Kurvenflug) nicht mehr erreichbar ist. Es muss dann automatisch auf das nächste Segment der geplanten Flugbahn weiter geschaltet werden.

Anforderung 6: Abfliegen eines S-Bahnsegments Der Autopilot soll in der Lage sein ein S-Segment aus der Bahnplanung abzufliegen. Dazu soll er entsprechende Vorgabewerte in eine Localizer-Interception Funktion eingespeist, und die Position des Flugzeugs relativ zum Endpunkt des S-Segments überwacht werden.

Anforderung 7: Localizer-Interception im Air-Frame Der Autopilot soll die Möglichkeit bieten nicht nur auf geradem Weg zu einem bestimmten Punkt zu fliegen, sondern er soll diesen auch in einer bestimmten Flugrichtung erreichen, so dass der Übergang in den nachfolgenden Kreisflug tangential erfolgen kann¹⁰.

Anforderung 8: Abfliegen eines L/R-Bahnsegments Der Autopilot soll in der Lage sein ein L- bzw. ein R-Segment aus der Bahnplanung abzufliegen. Dazu soll er entsprechende Vorgabewerte in die Circle-Fly Funktion einspeisen und die Position des Flugzeugs relativ zum Endpunkt des L/R-Segments überwachen.

Anforderung 9: Sequencing von Bahnprimitiven Der Autopilot soll in der Lage sein nacheinander eine Sequenz von vorgegebenen Bahnprimitiven abzufliegen. Die Umschaltung zwischen den einzelnen Bahnprimitiven soll automatisch erfolgen. Wenn der Zielpunkt eines Bahnprimitivs nicht mehr erreichbar ist, so soll der Autopilot automatisch auf das nächste erreichbare Bahnprimitiv umschalten.

4.3.2 Anforderungen an die Bahnplanung

Für die Evaluation des Algorithmus ist es essentiell, dass eine Bahnplanung einfach, schnell und flexibel erfolgen kann, um verschiedene Szenarien durchzuspielen. Dazu ist es hilfreich nicht nur eine bestimmte, reale Landebahn auszuwählen und deren Konfiguration als Ziel zu übernehmen, sondern zusätzlich die Möglichkeit zu haben eine fiktive Landebahn relativ zur aktuellen Flugzeugposition zu definieren. Damit lassen sich wesentlich einfacher verschiedene Modifikationen prüfen, ohne dass das Flugzeug immer aufwändig auf bestimmte Startkoordinaten positioniert werden muss. Insbesondere für Experimente in verschiedenen Höhen ist der Einsatz solcher virtueller Landebahnen unumgänglich.

Anforderung 10: Bahnplanung zu einer relativ zur aktuellen Flugzeugposition gegebenen Konfiguration Die Bahnplanung soll nach Angabe einer relativen Position und Drehung eine zugehörige Zielkonfiguration K_T ermitteln und einen Dubins-Anflug mit passendem Höhenverlust nach dem oben beschriebenen Algorithmus dorthin berechnen.

¹⁰ Der ausgerichtete Strahl durch den Zielpunkt auf dem sich das Flugzeug idealerweise bewegt wird als Localizer bezeichnet. Siehe dazu auch Abbildung 5.8.

Neben der Definition virtueller Landebahnen soll es möglich sein auch reale Landefelder auszuwählen und Anflüge dorthin tatsächlich bis zum Ende durchzuführen. Diese Auswahl sollte darauf vorbereitet sein die Daten nicht nur aus einer vorgegebene Liste zu wählen, sondern auch aus einer Datenbank wie in [Eckstein u. a., 2018] skizziert abzufragen.

Anforderung 11: Bahnplanung zu Koordinaten im WGS84-System Die Bahnplanung soll nach Angabe einer Zielkonfiguration $K_T = (P_T, \psi_T)$ in WGS84-koordinaten einen Dubins-Anflug mit passendem Höhenverlust nach dem oben beschriebenen Algorithmus zu berechnen.

Für das Experimentieren hat es sich als sehr praktisch erwiesen eine Funktion zu implementieren, die eine quasi kontinuierliche Neuberechnung der Bahnplanung ermöglicht. Damit kann immer ein definierter Zustand eingestellt werden, selbst wenn sich das Flugzeug im X-Plane Simulator weiter bewegt während in der Autopilot-Software noch Einstellungen gemacht werden.

Anforderung 12: Kontinuierliche Berechnung von geplanten Anflügen Die Bahnplanung soll eine Möglichkeit bieten einen Dubins Anflug zu einer relativ oder absolut im WGS84-System angegebenen Zielkonfiguration (quasi-) kontinuierlich zu wiederholen und so eine stets aktualisierte Bahnplanung bereitzuhalten, auch wenn der Simulator während vorgenommenen Einstellungen weiter läuft.

4.3.3 *Anforderungen an die Visualisierung und das Logging*

Das Visualisierungs- und vor allem das Loggingmodul sind notwendig, um die Plausibilität der gefundenen Ergebnisse zu überprüfen und zu dokumentieren. Mittels der Visualisierung ist eine qualitative Beurteilung der Autopilotfunktion und der Bahnplanung möglich. Mit Hilfe ausführlicher Logfiles kann eine weitergehende quantitative Auswertung vorgenommen werden.

Anforderung 13: Visualisierung der Flugbahn und der Bahnplanung Es soll eine Möglichkeit vorhanden sein eine Flugbahn und die Bahnplanung in einer Projektion auf einer Karte darzustellen. Die Darstellung der Flugbahn soll dabei mit einer ausreichenden Rate aktualisiert werden. Bei der Darstellung ist die Umrechnung der Geometrien vom Air-Frame in den Earth-Frame zu berücksichtigen.

Anforderung 14: Logging der relevanten Flugdaten Es soll eine Möglichkeit bestehen alle relevanten Flugdaten und Parameter in eine Datei auszugeben. Für eine spätere Auswertung ist ein geeignetes Datenformat vorzusehen.

Es ist offensichtlich, dass diese Anforderungen nicht hinreichend sind, um ein vollständiges Autopilotensystem konkret zu implementieren. Es handelt sich dabei lediglich um die wichtigsten Forderungen, die einen wesentlichen Einfluss auf die Implemen-

tierung hatten. Weitere Implementierungsdetails sind den Quellen und den darin enthaltenen Kommentaren zu entnehmen.

5 IMPLEMENTIERUNGSDetails AUSGEWÄHLTER KOMPONENTEN

In diesem Kapitel sollen einige ausgewählte Aspekte der SW-Architektur und der Implementierung genauer erklärt werden. Auch hier gilt, dass keine vollständige Beschreibung der gesamten erstellten SW gegeben werden kann, sondern lediglich besonders relevante oder interessante Teile herausgegriffen werden.

5.1 Implementierung des Bahnplanungsmoduls

Für das ELA-System ist eine Bahnplanungskomponente notwendig, die verlängerte Dubins-Pfade im Air-Frame berechnet und dem Autopilot als Vorgabe zur Verfügung stellt. Die Bahnplanung beschränkt sich auf die Fälle von LSLS und RSRS Anflügen, wie sie in der Algorithmenskizze in Abschnitt 3.3 beschrieben sind.

In der vorliegenden Implementierung ist das Bahnplanungsmodul Teil der Autopilot-Software, kann jedoch leicht durch ein externes Modul mit erweiterter Funktionalität ersetzt werden. Zum Beispiel findet aktuell keine Windkorrektur bei der Bahnplanung statt. Entsprechende Korrekturen können nach Auswertung der in dieser Arbeit durchgeführten Experimente zur Vorhersage der Flugdauern eingebracht und überprüft werden.

CHARAKTERISTISCHE DATEN ZUR BAHNPLANUNG Zur Berechnung eines verlängerter Dubins-Pfades im Air-Frame werden folgende Angaben benötigt:

- Startpunkt S , Endpunkt T
- Startheading ψ_S und Endheading ψ_T
- Gleitwinkel Geradeausflug γ_s und Gleitwinkel Kreisflug γ_c
- Gewünschter Höhenverlust Δh_{soll}
- Kreisradius r
- Pfadtyp: *LSLS* oder *RSRS*

Mit diesen Angaben lässt sich der gewünschte Anflug bestimmen, oder aber feststellen, dass ein solcher Anflug nicht möglich ist, da der gewünschte Höhenverlust Δh nicht realisierbar ist. (Siehe dazu [Klein u. a., 2018, Figure 8]). Ein verlängerter Dubins-Pfad im Air-Frame wird durch folgende Punkte und Werte eindeutig charakterisiert:

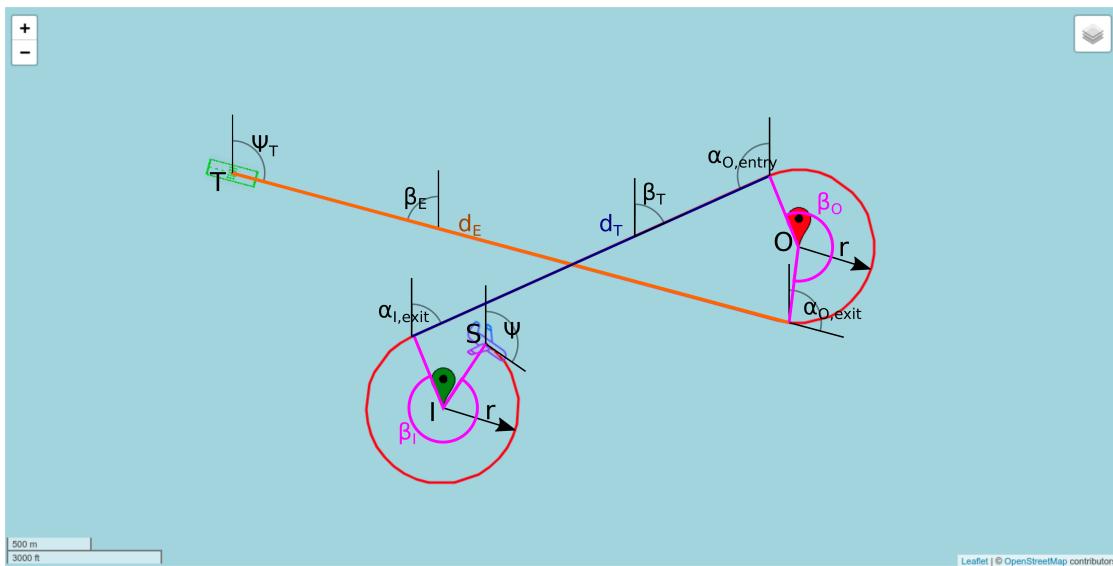


Abbildung 5.1: Die charakteristischen Punkte und Winkel eines verlängerten RSRS-Dubins-Anflugs

- Startpunkt S, Endpunkt T
- Kreismittelpunkt „In-Circle“ I, Kreismittelpunkt „Out-Circle“ O, Kreisradius r
- Kreiseingangswinkel $\alpha_{I,\text{entry}}$, $\alpha_{O,\text{entry}}$, Kreisausgangswinkel $\alpha_{I,\text{exit}}$, $\alpha_{O,\text{exit}}$
- Kreisrotation β_I , β_O ¹¹
- Pfadtyp entweder LSLS oder RSRS

Aus diesen charakteristischen Werten lassen sich weitere wichtige Daten ableiten:

- Der Gesamtdrehwinkel $\beta = \beta_I + \beta_O$
- Der Winkel β_T des Tangentialsegments d_T und der Winkel $\beta_E \equiv \psi_T - 180^\circ$ des Endsegments d_E
- Die Flugdistanzen auf den einzelnen Segmenten d_I , d_O , d_T und d_E
- Die Höhenverluste auf den einzelnen Segmenten Δh_I , Δh_O , Δh_T und Δh_E

Die Implementierung der Bahnplanung stützt sich auf diese Werte, die im Folgenden verwendet werden und deren Berechnung detaillierter erläutert wird. Die genannten Werte sind zur Verdeutlichung in Abbildung 5.1 veranschaulicht.

¹¹ Der Gesamtdrehwinkel kann $> 360^\circ$ werden und lässt sich daher nicht eindeutig aus $\alpha_{\text{exit}} - \alpha_{\text{entry}}$ berechnen.

VORBEREITUNG Die Positionen der relevanten Punkte im Dubins-Pfad werden in WGS84-Erdkoordinaten gespeichert. Damit wird hauptsächlich die Visualisierung erleichtert. Die Berechnungen sowohl für die Bahnplanung als auch für die späteren Autopilot-Funktionen finden dagegen überwiegend in kartesischen Koordinaten statt. Da die relevanten Distanzen hinreichend kurz sind, kann ein „flat earth“ Modell (siehe [Allerton, 2009, chapter 1.4.12]) verwendet werden, bei dem die Positionen auf eine Tangentialebene der Erde projiziert werden. Zur Umrechnung zwischen den beiden Koordinatensystemen werden die Opensource Bibliothek **GeographicLib**¹² und einige eigene Hilfsfunktionen verwendet.

Vor der Berechnung des gewünschten Anflugs werden zunächst die Positionen in kartesische Koordinaten umgerechnet. Der Algorithmusskizze aus Abschnitt 3.3 folgend wird als Ursprung des kartesischen Koordinatensystems der Zielpunkt T verwendet und alle Winkel werden so gedreht, dass für die so transformierte Endheading $\hat{\psi}_T = 270^\circ$ gilt. Diese Koordinatentransformation muss am Ende der Berechnung wieder rückgängig gemacht werden, erleichtert aber viele Zwischenschritte der Berechnung enorm. *In der folgenden Implementierungsbeschreibung der Bahnplanung wird nicht explizit erwähnt, ob es sich um Originalkoordinaten und -winkel handelt oder um transformierte Werte. Das geht aus dem Zusammenhang eindeutig hervor.*

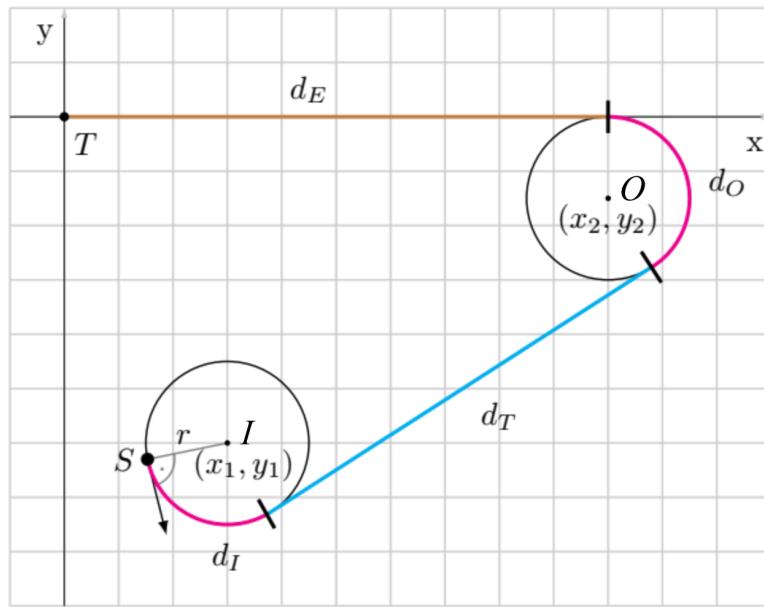


Abbildung 5.2: Der zu planende Dubins-Pfad in der kartesischen Ebene
(nach [Klein u. a., 2018, fig. 4])

Nach dieser vorbereitenden Transformation liegt die zu planende Bahn wie in Abbildung 5.2 dargestellt in der kartesischen Ebene.

¹² Bereitgestellt unter <https://geographiclib.sourceforge.io/html/index.html>

Die Berechnung der verlängerten Dubins-Pfade wurde zweistufig implementiert. Im ersten Schritt wird der Dubins-Pfad vom Typ LSL bzw. RSR von S nach T berechnet. Der gewünschte Höhenverlust Δh wird noch nicht über das Einfügen eines Final Approach d_E korrekt eingestellt. Wenn sich herausstellt, dass dieser unverlängerte Dubins-Pfad zulässig ist, aber nicht ausreichend Höhenverlust generiert, wird im zweiten Schritt die noch einzufügende Länge d_E bestimmt, so dass genau der Höhenverlust Δh während des Anflugs aufgezehrt wird.

BERECHNUNG DER KREISMITTELPUNKTE Die Berechnung der Kreismittelpunkte ist direkt möglich: Der Kreismittelpunkt I des Eingangskreises liegt genau mit dem Abstand r links bzw. rechts von der aktuellen Flugzeugposition. Es muss lediglich von der Startposition S ein Vektor in Flugrichtung mit Länge r um -90° (nach links) bzw. $+90^\circ$ (nach rechts) rotiert werden, um I zu bestimmen.

Der Mittelpunkt des Ausgangskreises ist aufgrund der vorbereitenden Winkel-Transformation noch leichter zu bestimmen: O liegt direkt unterhalb bzw. oberhalb des Ursprungs auf der Position $O = (x_2 = 0, y_2 = -r)$ bzw. $O = (x_2 = 0, y_2 = +r)$.

BERECHNUNG DER RELEVANTEN WINDEL Für den Höhenverlust im Kreisflug ist der Gesamtdrehwinkel β relevant, den das Flugzeug insgesamt auf den Kreissegmenten überstreicht.

Das gerade Segment d_T , welches die Kreissegmente verbindet, ist eine Tangente an die beiden durch ihre Mittelpunkte I und O und Radius r definierten Kreise. Im Fall von gleichläufigen Kreisen liegt die gesuchte Tangente parallel zu der Verbindungsline der beiden Kreismittelpunkte. Sie lässt sich konstruieren, indem diese Verbindungsline um den Abstand r parallel verschoben wird, bis sie zur Tangente wird.

Der Winkel β_T des Tangentialsegments und damit der gesuchten Tangente d_T lässt sich aus den Koordinaten der Kreismittelpunkte bestimmen zu $\beta_T = \text{atan}_2((x_2 - x_1), (y_2 - y_1))$. Identisch damit sind der Kreisausgangswinkel α_I und der Kreiseingangswinkel α_O . Es gilt $\alpha_{I,\text{exit}} = \beta_T$ und $\alpha_{O,\text{entry}} = \beta_T - 180^\circ$. Für den Kreiseingangswinkel α_I gilt $\alpha_{I,\text{entry}} = \psi_S$ und für den Kreisausgangswinkel α_O gilt $\alpha_{O,\text{exit}} = \psi_T$

Das Flugzeug muss den Gesamtdrehwinkel β in zwei Schritten überstreichen. Zunächst muss es sich auf dem Eingangskreis um den Winkel β_I in Richtung der Tangente drehen, bevor es sich auf dem Ausgangskreis um den Winkel β_O in seine Endausrichtung dreht. Aufgrund dieser zweischrittigen Drehung kann der Gesamtdrehwinkel $\beta = \beta_I + \beta_O$ nicht direkt aus der Differenz der Endausrichtung ψ_T und der Startausrichtung ψ berechnet werden. Je nach Ausgangskonfiguration kann es vorkommen, dass ein zusätzlicher Kreis von 360° geflogen werden muss.

Bei einer direkten Berechnung von β muss eine Fallunterscheidung berücksichtigt werden:

$$\beta = \begin{cases} \psi_T - \psi + 360^\circ & \text{für } \beta_T - 90^\circ \leq \psi < \beta_T \\ \psi_T - \psi & \text{sonst} \end{cases} \quad (5.1)$$

Durch die Implementierung als zweischrittige Drehung wird diese Fallunterscheidung implizit vorgenommen und es werden gleichzeitig die Werte der Kreiseingangs- und Kreisausgangswinkel berechnet.

BERECHNUNG DER HÖHENVERLUSTE Der Höhenverluste entlang des berechneten kürzesten Dubins-Pfads kann aus den Längen der einzelnen Segmente zusammen mit dem jeweils gültigen Gleitwinkel berechnet werden.

Die Länge des Tangentialstückes $d_{T,short}$ im unverlängerten Pfad entspricht genau dem Abstand der beiden Kreismittelpunkte. Mit den Setzungen

$$\Delta x := (x_2 - x_1) \stackrel{x_2=0}{=} -x_1 \quad \text{und} \quad \Delta y := (y_2 - y_1) \stackrel{y_2=\mp r}{=} (\mp r - y_1) \quad (5.2)$$

kann man einfach schreiben

$$d_T = \sqrt{\Delta x^2 + \Delta y^2} \quad (5.3)$$

Die Länge der Kreissegmente ergibt sich direkt aus den Kreiswinkeln und dem Radius:

$$d_I = r\beta_I \quad \text{und} \quad d_O = r\beta_O \quad (5.4)$$

Die Höhenverluste der einzelnen Segmente ergeben damit

$$\begin{aligned} \Delta h_I &= d_I \tan(\gamma_c) = r\beta_I \tan(\gamma_c) \\ \Delta h_O &= d_O \tan(\gamma_c) = r\beta_O \tan(\gamma_c) \\ \Delta h_{T,short} &= d_T \tan(\gamma_s) = \sqrt{(x_1)^2 + (y_1 \pm r)^2} \tan(\gamma_s) \\ \Delta h_{ist} &= \Delta h_I + \Delta h_O + \Delta h_T \end{aligned} \quad (5.5)$$

Für den unverlängerten gleichläufigen Dubins-Pfad LSL bzw. RSR sind nun alle Parameter bestimmt. Es kann geprüft werden, ob der gewünschte Zielpunkt überhaupt erreichbar ist, oder ob die verbleibende Residualhöhe des Flugzeugs nicht ausreicht, um zum Notlandefeld zu gelangen.

Für einen erfolgversprechenden Anflug muss gelten $\Delta h_{soll} \geq \Delta h_{ist}$.

VERLÄNGERUNG DES ANFLUGS In den meisten Fällen wird die Bedingung $\Delta h_{soll} > \Delta h_{ist}$ erfüllt sein, so dass der geplanten Bahn noch ein gerades Endanflugsegment d_E hinzugefügt werden muss, um die überschüssige Höhe $\Delta h_{additional} := \Delta h_{soll} - \Delta h_{ist}$ abzubauen. Dazu wird, wie in Abbildung 3.3 zu sehen ist, der Kreismittelpunkt O nach rechts vom Zielpunkt T weg verschoben. Dabei verändert sich auch die Länge des Tangentialsegments von $d_{T,short}$ zu $d_{T,long}$. Es gilt nicht notwendigerweise $d_{T,short} < d_{T,long}$.

Die Länge d_E des eingefügten Final Approach Segments ist so zu bestimmen dass gerade gilt

$$\Delta h_{soll} \stackrel{!}{=} \Delta h_{ist} = \Delta h_I + \Delta h_O + \Delta h_{T,long} + \Delta h_E \quad (5.6)$$

Der Gesamtdrehwinkel β bei Verschiebung des Ausgangskreises ändert sich in der Regel nicht. Die Länge der Kreissegmente und damit die Summe der Höhenverluste im Kreisflug $\Delta h_{circle} = \Delta h_I + \Delta h_O$ bleibt daher konstant. Damit lässt sich die im Geradeausflug zu vernichtende Höhe $\Delta h_{straight}$ und damit die Summe $d_{T,long} + d_E$ berechnen zu

$$\begin{aligned}\Delta h_{straight} &= \Delta h_{soll} - \Delta h_{circle} = \Delta h_{soll} - r\beta \tan(\gamma_c) \\ d_{T,long} + d_E &= \frac{\Delta h_{straight}}{\tan(\gamma_s)}\end{aligned}\quad (5.7)$$

Für die Koordinaten des Ausgangskreises gilt nach Einfügen des Final-Approach $O = (x_2 = d_E, y_2 = \mp r)$. Die neue Länge des Tangentialsegments $d_{T,long}$ ist durch den neuen Abstand der Kreismittelpunkte definiert. Für die Längensumme $d_{straight}$ der beiden geraden Segmente gilt damit

$$d_{straight} := d_T + d_E = d_E + \sqrt{(x_1 - d_E)^2 + (y_1 \pm r)^2} \stackrel{!}{=} \frac{\Delta h_{straight}}{\tan(\gamma_s)} \quad (5.8)$$

Die Verlängerung des Anflugs kann mit den Setzungen Δx und Δy des unverlängerten Anflugs aus Gleichung (5.2) berechnet werden zu:¹³

$$d_E = -\frac{\Delta y^2 + \Delta x^2 - d_{straight}^2}{2(d_{straight} + \Delta x)} \quad (5.9)$$

Mit den neuen Kreismittelpunkten und dem neu eingefügten Final-Approach Segment ist ein neuer, verlängerter Dubins-Pfad der Form *LSLS* bzw. *RSRS* definiert.

ÜBERPRÜFUNG DES VERLÄNGERTEN ANFLUGS Nach Konstruktion des verlängerten Dubins-Pfades sollte man erwarten, dass gilt: $\Delta h_{soll} = \Delta h_{ist}$.

Das ist jedoch nicht immer der Fall, da durch die Verschiebung des Kreismittelpunktes O auch der Winkel des Tangentialsegments $\beta_{T,long}$ verändert wird. Es kann bei dieser Änderung vorkommen, dass sich der Ausgang der Fallunterscheidung (5.1) ändert und der Gesamtdrehwinkel β_{long} im verlängerten Anflug um $\pm 360^\circ$ vom Drehwinkel β des kürzesten Dubins-Pfads abweicht.

Es muss daher nach Berechnung des verlängerten Pfades überprüft werden, ob die Annahme $\beta_{long} = \beta$ bezüglich des zu durchfliegenden Winkels im Kreisflug wirklich gilt.

Im algorithmisch einfacher zu behandelnden Fall gilt $\beta_{long} = \beta - 360^\circ$. Der verlängerte Anflug enthält eine vollständige Drehung weniger als der kürzeste Dubins-Pfad. Das lässt sich einfach korrigieren indem im verlängerten Anflug der Eingangskreis einmal öfter durchflogen wird, um wieder auf den ursprünglichen Gesamtdrehwinkel β und damit den ursprünglichen Höhenverlust $\Delta h_{circle} = \Delta h_I + \Delta h_O$ im Kreisflug zu kommen.

¹³ Dies ist das gleiche Ergebnis wie in [Klein u. a., 2018, eq. 6], jedoch schon so umgeformt, dass es leicht in Code umgesetzt werden kann.

Etwas komplizierter liegt der Fall, wenn gilt $\beta_{\text{long}} = \beta + 360^\circ$. Der verlängerte Anflug enthält eine vollständige Drehung mehr als der unverlängerte Dubins-Pfad. Damit ist der gesamte Höhenverlust nicht mehr, wie angenommen $\Delta h_{\text{ist}} = \Delta h_{\text{straight}} + \beta r \tan(\gamma_c)$, sondern es gilt $\Delta h_{\text{ist}} = \Delta h_{\text{straight}} + (\beta + 2\pi)r \tan(\gamma_c)$.

Falls für die auf der Verlängerung des Dubins-Pfades zusätzlich vernichtete Höhe gilt $\Delta h_{\text{additional}} \geq 2\pi r \tan(\gamma_c)$, so lohnt es sich, einen weiteren Berechnungsversuch durchzuführen, dessen zusätzlicher Höhenverlust auf $(\Delta h_{\text{additional}} - 2\pi r \tan(\gamma_c))$ reduziert wird. Wenn auch bei dieser reduzierten Verlängerung gilt $\beta_{\text{long}} = \beta + 360^\circ$, so ist dieser „reduziert verlängerte“ Dubins Pfad gültig. Der zusätzliche Kreis trägt gerade den bei der Berechnung der Verlängerung herausgenommenen Höhenverlust von $2\pi r \tan(\gamma_c)$ bei.

Es gibt jedoch auch Notlandefelder, die trotz eigentlich ausreichender Residualhöhe des Flugzeugs unerreichbar sind wie das in Abbildung 5.3 dargestellt wird. Häufig ist es dann ausreichend die Drehrichtung des Anflugs zu ändern um einen gültigen Anflug für das ELF zu erhalten.

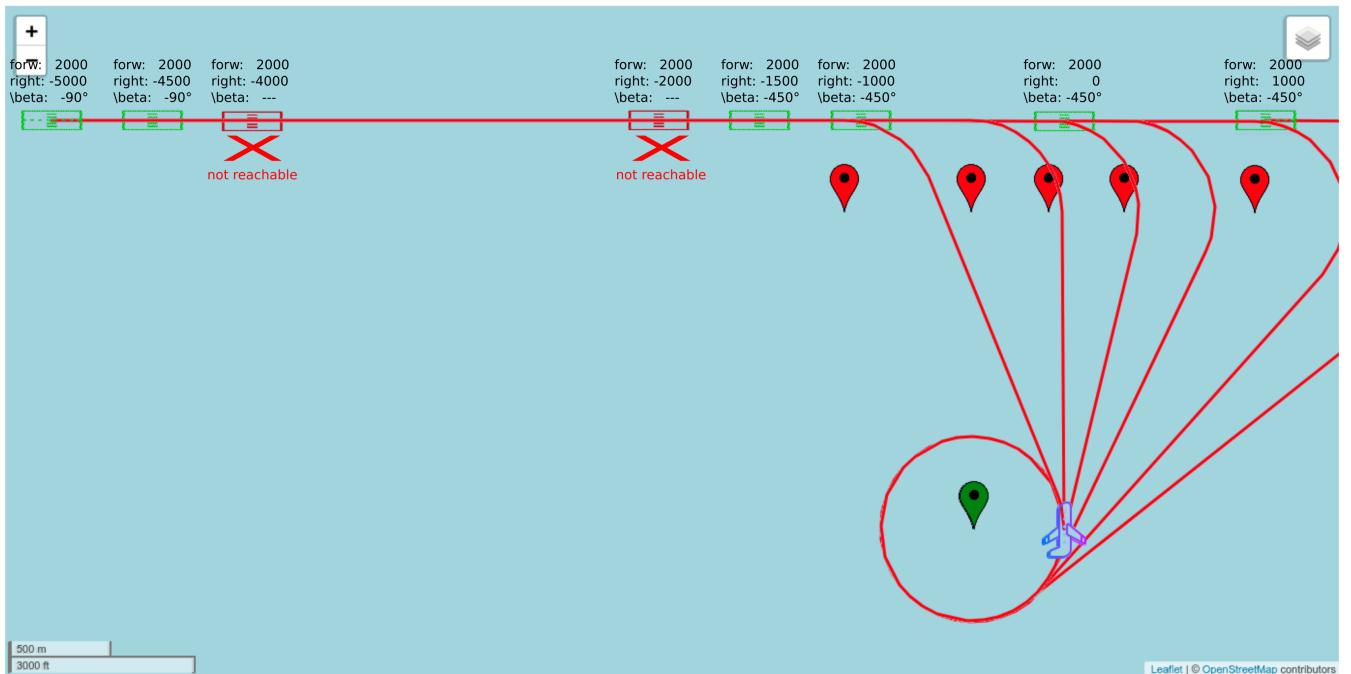


Abbildung 5.3: Darstellung erreichbarer und nicht erreichbarer ELF. Der Abstand zum Flugzeug beträgt $\Delta_{\text{forward}} = 2000\text{m}$, $\Delta_{\text{right}} = [-5000\text{m} \dots 1000\text{m}]$ und $\Delta h_{\text{soll}} = -600\text{m}$. ELF's im Bereich von $\Delta_{\text{right}} \approx [-4000\text{m} \dots -2000\text{m}]$ sind nicht auf einem LSLS-Pfad erreichbar.

Trotz der etwas unübersichtlichen Korrekturen berechnet die beschriebene Implementierung in den allermeisten Fällen einen gültigen Dubins-Anflug im Air-Frame.

Vor der Übergabe an den Autopiloten werden die vorbereitenden Transformationen wieder rückgängig gemacht und die Koordinaten des kartesischen Koordinatensystems werden in erdgebundene WGS84-Koordinaten umgewandelt, um die Visualisierung und den Abgleich mit X-Plane zu erleichtern. Die aufgrund von Wind notwendigen Korrekturen des Air-Frame gegenüber dem Earth-Frame werden später im Autopiloten während der Bahnregelung durchgeführt.

5.2 Implementierung der Regelkreise

Zur Realisierung des Autopiloten sind einige Regelkreise zu kontrollieren, um die Fluglage stabil zu halten und die Flugparameter im gewünschten Bereich zu halten. Wie schon in Abschnitt 4.2.2 erwähnt, sind Regelkreise für folgende Funktionen notwendig:

- Sink Control zur Regelung des Gleitwinkels
- Roll Controll zur Regelung des Rollwinkels zur Steuerung des Kurvenflugs
- Heading Hold zur Regelung eines Geradeausflugs in eine bestimmte Richtung
- Localizer Interception zum Anflug eines Punktes in einer vorgegebenen Richtung
- Circle Flight zum Kreisflugs um einen Mittelpunkt mit vorgegebenem Radius

Auf das Flugzeug wirken neben den Kontrollflächen (Höhen- und Querruder) vielerlei externe Störungen ein. Um eine bestimmte Fluglage zu erreichen müssen die Kontrollflächen so gesteuert werden, dass dem Einfluss dieser externen Störungen entgegen gewirkt wird und die Flugparameter die vorgegebenen Sollwerte erreichen. Dies ist die Aufgabe eines Regelkreises, dessen Grundstruktur in Abbildung 5.4 noch einmal gezeigt wird.

GRUNDLAGEN REGELKREIS Der zu regelnde Flugparameter $y(t)$ wird laufend gemessen und mit dem vorgegebenen Sollwert $w(t)$ für diesen Parameter verglichen. Dadurch wird die Regeldifferenz $\tilde{e}(t)$ ermittelt, die als Eingangsgröße für den Regler dient. Der Regler berechnet aus der aktuellen und eventuell früheren Reglerdifferenz ein geeignetes Stellsignal, das dem Stellglied zugeführt wird, um auf das zu regelnde System einzuwirken. Neben dieser gewollten Einwirkung kommt es zwangsläufig auch zu externen Störungen $z(t)$, die auch auf das System wirken. Das zu regelnde System selber bezeichnet man als Regelstrecke. Ein solches rückgekoppeltes System aus Regler, Stellglied, Regelstrecke und Sensorik wird als Regelkreis bezeichnet.

Mit Hilfe von Regelkreisen ist es möglich die Ausgangsparameter von Regelstrecken auf gewünschte Sollwerte zu bringen und sie dort zu halten, selbst dann, wenn

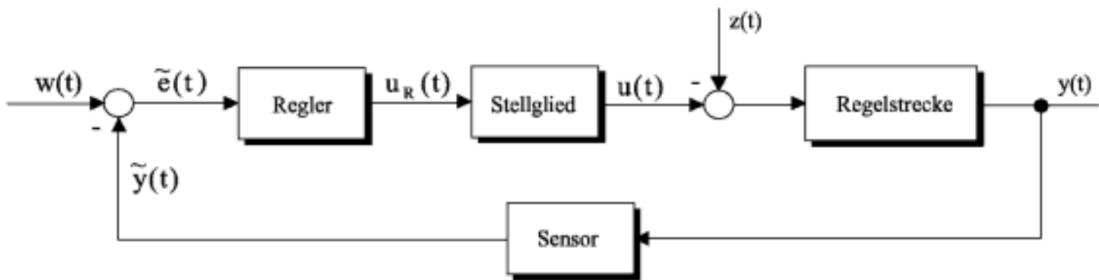


Abbildung 5.4: Grundstruktur eines Regelkreises
(aus [Berger, 2001, fig. 1.1])

das Übertragungsverhalten der Regelstrecke und das Auftreten von Störgrößen nicht bekannt ist. Da es sich bei den Regelstrecken im Flugzeug um LTI-Systeme (vgl. Definition in [Berger, 2001, p. 36]) handelt, ist ein PID-Regler (Proportional, Integral and Derivative) als Universallösung eine gute Wahl. Dies wird auch in [Allerton, 2009, p. 163 ff.] nahegelegt.

GRUNDLAGEN PID-REGLER Der PID-Regler berücksichtigt bei der Berechnung des Stellsignals sowohl die Größe des aktuellen Fehlers (*proportional*), das Integral der bisher aufgetretenen Fehler (*integral*) sowie die Änderungsgeschwindigkeit (*derivative*) des Fehlers. Die Gewichtungsfaktoren dieser drei Anteile K_p , K_I und K_d werden als Reglerparameter bezeichnet. Diese Reglerparameter müssen gut auf die Regelstrecke abgestimmt werden, weil der Regelkreis ansonsten instabil werden kann oder die gewünschten Werte der Ausgangsparameter nicht erreicht werden.

In der einschlägigen Fachliteratur zur Regelungstechnik wird eine Vielzahl von Methoden zur Bestimmung und Optimierung der Reglerparameter angegeben, von denen zumindest einige eine gute Kenntnis der Regelstrecke erfordern. Glücklicherweise ist es bei den meisten Regelkreisen aber auch möglich mit empirisch ermittelten Reglerparametern ein ausreichend gutes Regelverhalten zu erreichen. Auch Faustformel-Verfahren zur empirischen Ermittlung der Parameter werden in der Literatur ausführlich beschrieben.

Da die Regelstrecke „Flugzeug“ sehr komplex und für jeden Flugzeugtyp anders ist werden auch in dieser Arbeit die Reglerparameter empirisch ermittelt. Die GUI des Experimentiersystems bietet für jeden der vier genannten Regelkreise die Möglichkeit die Parameter K_p , K_I und K_d interaktiv einzustellen. Durch den Einsatz eines Simulators ist eine solche interaktive Herangehensweise problemlos möglich, da auch bei instabilen Zwischenständen keine Gefahr droht.

Für die Einschätzung der Parameter hat sich Tabelle 5.1 (aus Allerton [2009]) als sehr nützlich erwiesen.

Control term	Rise time	Overshoot	Settling time	Steady-state error
K_p	Decrease	Increase	Small change	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_d	Small change	Decrease	Decrease	Small change

Tabelle 5.1: Effekt der PID Reglerparameter (aus [Allerton, 2009, tab. 4.1])

Für die Implementierung des PID-Reglers selber wurde die Bibliothek `PID Controller` von **Trent Cleghorn** als Basis verwendet und validiert.¹⁴

Obwohl diese Implementierung schon viele Fallstricke einer realen Implementierung des in der Theorie sehr einfachen PID-Konzepts umschift, mussten noch einige Modifikationen eingebbracht werden:

- Externes Rücksetzen des Integrators
- Übergabe angepasster Funktionen zur Messung des aktuellen Ist-Wertes; Signal-konditionierung
- Übergabe angepasster Funktionen zur Berechnung der Reglerdifferenz
- Übergabe angepasster Funktionen zur Rückgabe der Stellwerte

Die Notwendigkeit dafür wird später deutlich, wenn die einzelnen Regelkreise beschrieben werden, die auf die Verbesserungen angewiesen sind.

Um für bestimmte Aufgaben speziell angepasste Funktionen nutzen zu können, wurden C++11-Funktionsobjekte eingesetzt: In der generischen PID-Regler-Implementierung wird ein Funktionsobjekt für die fraglichen Funktionen als Membervariable angelegt und mit der Standard-Funktion vorbelegt. Diese Variable kann bei Bedarf für bestimmte Regler mit einer angepassten Lambda-Funktion überschrieben werden. Diese angepasste Funktion ersetzt die Standard-Implementierung.

Nach diesen Grundlagen zu Reglern im Allgemeinen werden nachfolgend die tatsächlichen Regelkreise im Autopilot beschrieben und es wird angegeben, wie die konkreten Regler implementiert sind.

SINK CONTROL ZUR REGELUNG DES GLEITWINKELS [Abbildung 5.5] Dieser Regler kontrolliert das Höhenruder, um den Gleitwinkel auf einem vorgegebenen Wert zu halten. Da der Gleitwinkel in X-Plane nicht direkt als Messwert verfügbar ist,¹⁵ wur-

¹⁴ Bereitgestellt unter https://github.com/tcleg/PID_Controller und ausführlich dokumentiert unter <http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>.

¹⁵ Der in X-Plane verfügbare Wert `sim/flightmodel/position/vpath` gibt den auf den Earth-Frame bezogenen Winkel $\arctan\left(\frac{V_{GS}}{V_{sink}}\right)$ zurück, der für eine Regelung auf „best glide“ im Air-Frame unter Windeinfluss nicht brauchbar ist.

de die Funktion `getMeasurement` des generischen PID-Reglers mit einer angepassten Funktion überschrieben, die den Gleitwinkel aus V_{TAS} und V_{sink} berechnet.

$$\gamma = \arctan\left(\frac{1}{E}\right) = \arctan\left(\frac{V_{TAS}}{V_{sink}}\right) \quad (5.10)$$

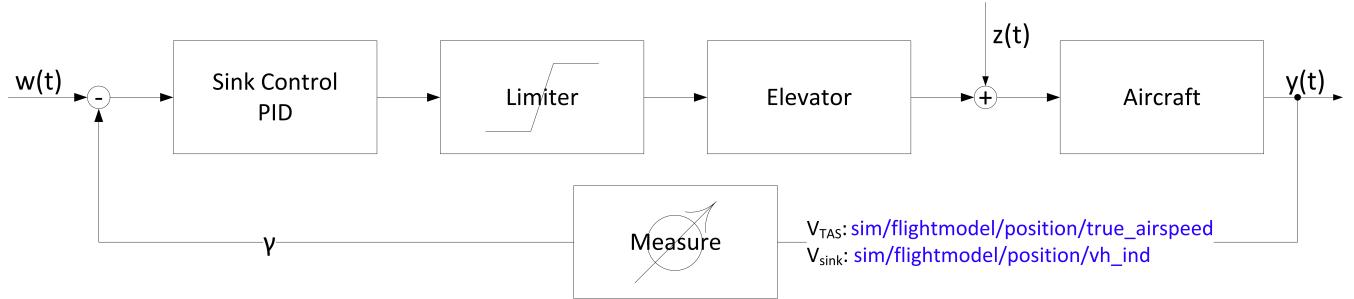


Abbildung 5.5: Der PID-Regler zur Regelung des Gleitwinkels

Der Sollwert des Sink Control Reglers wird auf einen von zwei Vorgabewerten gesetzt: entweder γ_s im Geradeausflug oder auf γ_c im Kreisflug.¹⁶

ROLL CONTROL ZUR REGELUNG DES ROLLWINKELS [Abbildung 5.6] Dieser Regler kontrolliert das Querruder und damit den Rollwinkel des Flugzeugs. Bei der Parametrierung des Reglers ergibt sich der Wert $K_d = 0$. Damit degeneriert der PID-Regler zu einem reinen PI-Regler.

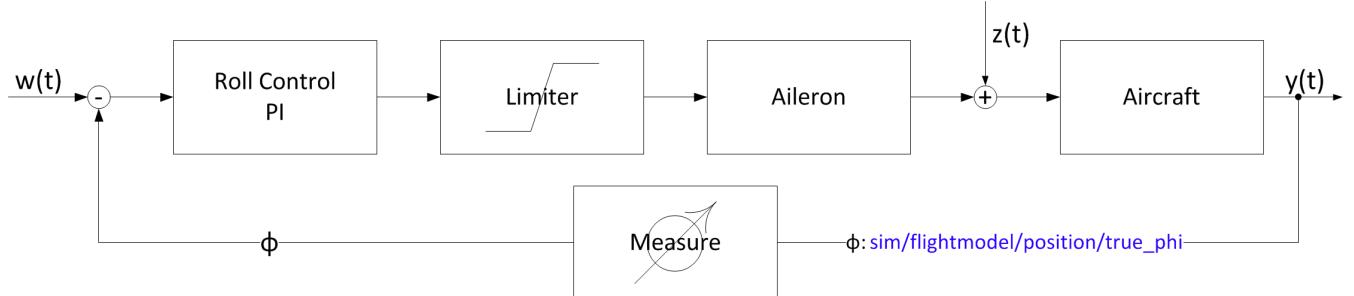


Abbildung 5.6: Der PI-Regler zur Regelung des Rollwinkels

¹⁶ Während der Programmierung wurde damit experimentiert den Sollwert γ dynamisch an den aktuelle Rollwinkel anzupassen. Dazu wurde eine Lookuptable mit Werten aus dem Best-Glide Hodographen benutzt. Es hat sich aber gezeigt, dass diese dynamische Anpassung komplexe Rückwirkungen nach sich zieht, die den Regler sehr „nervös“ werden lassen und daraufhin zu relativ starken Ruderausschlägen führen. Augenscheinlich geht durch eine solche „Überregelung“ mehr Energie und damit Gleitweg verloren als durch die dynamische Anpassung an den Best Glide gewonnen wird. Für reproduzierbare Experimente wurde daher entschieden nur zwei Vorgabewerte für den Gleitwinkel zu benutzen.

HEADING HOLD ZUR REGELUNG DES GERADEAUSFLUGS [Abbildung 5.7] Dieser Regler steuert das Flugzeug in eine bestimmte Himmelsrichtung ψ indem es den Kurvenflug kontrolliert. Als Stellgröße wird ein Sollwert für die Roll-Control berechnet und in diesen Regler eingespeist. Für die Berechnung der Reglerdifferenz muss die Funktion calculateError des generischen PID-Reglers überschrieben werden, um der 360° -Periodizität der Winkelmessung Rechnung zu tragen.

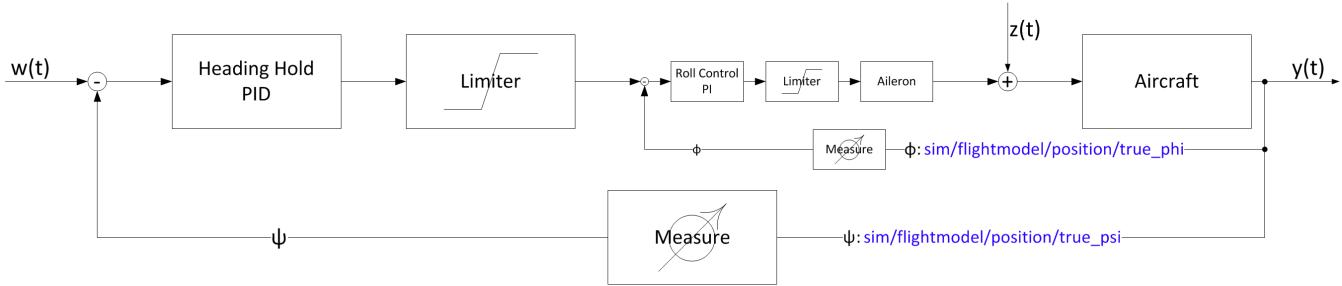


Abbildung 5.7: Der PID-Regler zur Regelung des Geradeausflugs. Prinzip der Reglerkaskadierung.

LOCALIZER INTERCEPTION [Abbildung 5.9] Für das Abfliegen der geraden Segmente d_T und d_E des geplanten Anflugs scheint es zunächst naheliegend, immer direkt auf den gewünschten Zielpunkt zuzufliegen. Das kann leicht über die Heading-Control realisiert werden, deren Sollwert im Verlauf des gesamten Anflugs auf die Richtung zwischen aktueller Position und Zielpunkt angepasst wird. Bei einer solch einfachen Regelung wird zwar der Punkt erreicht werden, jedoch kann die Flugrichtung bei Erreichen des Ziels erheblich von der vorberechneten Richtung abweichen.

Damit der Übergang in das sich anschließende Kreissegment tangential erfolgen kann muss eine solche Richtungsabweichung möglichst vermieden werden. Das Problem einen Punkt in einer bestimmten Richtung zu erreichen ist unter dem Begriff „Localizer Interception“ bekannt. Ein einfacher Regelkreis wird in [Allerton, 2009, chap. 4.11] beschrieben und soll hier anhand der dort angegebenen Abbildung 5.8 erklärt werden.

Es wird versucht die ideale Anflugline (den Localizer) schon vor dem Zielpunkt B zu erreichen. Dazu wird ein virtueller Punkt P berechnet, auf den das Flugzeug zusteuert, so dass die Localizer Linie geschnitten wird. Es wird permanent die Richtung ψ_{diff} des direkten Weges von der aktuellen Position zur Zielposition bestimmt und mit der gewünschten Richtung im Ziel ψ_{target} verglichen. Der Sollwert des Heading-Hold Reglers wird auf den Wert $\psi_{\text{target}} - K_P(\psi_{\text{target}} - \psi_{\text{diff}})$ eingestellt, wobei sich für den Wert K_P ein Wert von 6 als praktikabel herausgestellt hat (siehe [Allerton, 2009, ch. 4.11]). Dieses Verfahren beschreibt einen einfachen P-Regler.

Für die Messung des Winkels ψ_{diff} im Air-Frame benötigt dieser Regler eine angepasste Messfunktion: Da die Flugzeugposition im Earth-Frame (in WGS84 Koordinaten)

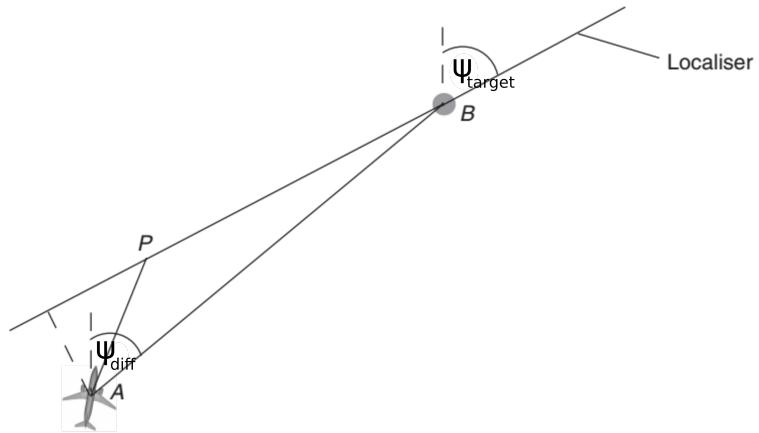


Abbildung 5.8: Prinzip der Localizer Aquisition
(aus [Allerton, 2009, fig. 4.24])

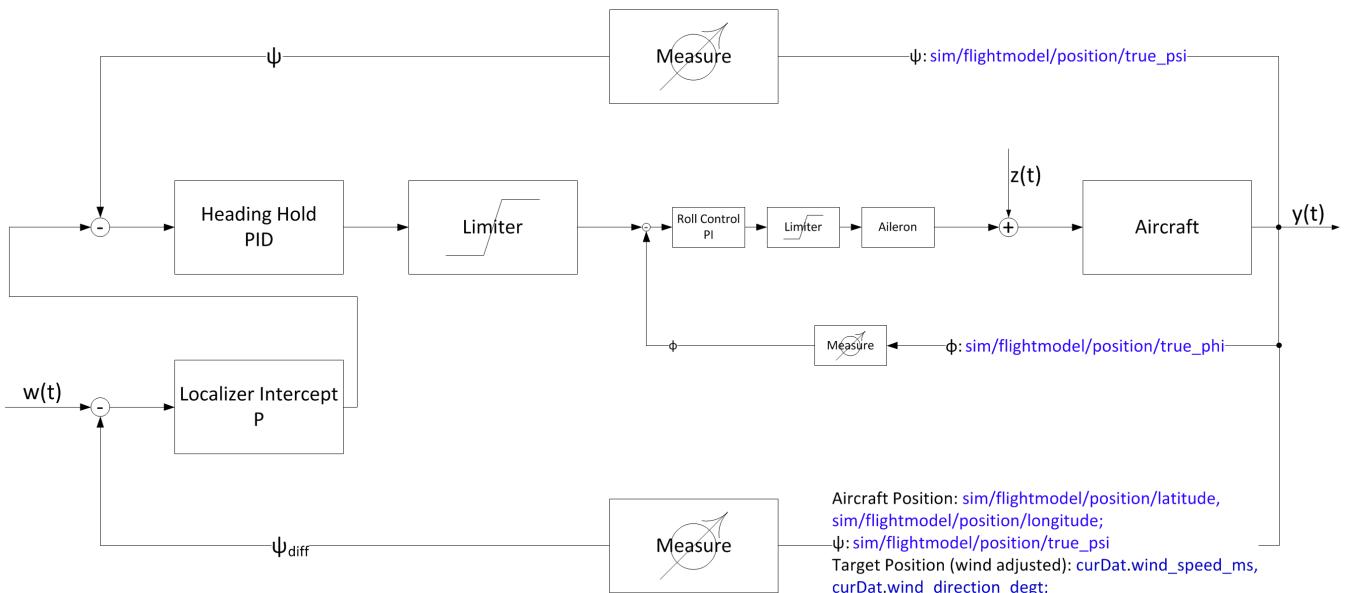


Abbildung 5.9: Der P-Regler zur Straight Segment Interception. Zusammenspiel mit dem Heading-Hold-Regler.

ten) gegeben ist, muss der Zielpunkt auch in den Earth-Frame transformiert werden. Die angepasste Messfunktion berechnet daher vor jeder Messung die durch den Wind hervorgerufene Verschiebung des Air-Frames gegen den Earth-Frame seit Beginn des Fluges und korrigiert die Winkelmessung zwischen Flugzeug und Zielpunkt entsprechend.

Dieser einfache P-Regler ist als Einziger nicht als generischer PID-Regler mit der genannten Bibliothek implementiert, sondern direkt in das Bahnverfolgungsmodul

integriert. Damit lässt sich der K_p -Parameter nicht interaktiv einstellen. Es hat sich gezeigt, dass der in der Literatur angegebene Wert sehr gut funktioniert.

CIRCLE-FLY FÜR DEN KREISFLUG [Abbildung 5.10] Dieser Regler soll das Flugzeug auf einen Kreisflug um einen im Air-Frame festen Mittelpunkt zwingen, dessen Radius vorgegeben werden kann.

Der Kreisflug selber wird dadurch erreicht, dass sowohl der Sollwert als auch die Stellgrößenbegrenzung des *Heading-Hold*-Reglers laufend überschrieben werden: Der Sollwert des *Heading-Hold* wird bei jedem Regleraufruf auf einen Wert gesetzt, der im Drehsinn des Kreises 45° vor der aktuellen Flugrichtung liegt. Der *Heading Hold*-Regler hat damit keine Chance die angefragte Richtung jemals zu erreichen obwohl er das Flugzeug maximal in den Kurvenflug bringt und dazu einen entsprechenden Rollwinkel bei der Roll-Control vorgibt. Neben der (nicht erreichbaren) Sollwertvorgabe wird im Kreisflug auch der maximale Ausgangsrollwinkel φ_{\max} der *Heading-Hold*-Regelung bei jedem Regleraufruf unter Berücksichtigung der aktuellen V_{TAS} angepasst so dass sich rein rechnerisch der gewünschte Kurvenradius ergibt.

$$\varphi_{\max} = \arctan \left(\frac{V_{TAS}^2}{g r} \right) \text{ mit } g \approx 9,81 \frac{\text{m}}{\text{s}^2} \quad (5.11)$$

Da aber auch der Kurvenradius und die Geschwindigkeitsmessung Störungen unterliegen wird für die rückgekoppelte Regelung des Kurvenradius der Circle-Fly-Regler eingesetzt. Für die Messung des Radius im Air-Frame benötigt dieser Regler eine angepasste Messfunktion: Da die Flugzeugposition im Earth-Frame (in WGS84 Koordinaten) gegeben ist, muss der Kreismittelpunkt auch in den Earth-Frame transformiert werden. Die angepasste Messfunktion berechnet daher vor jeder Messung die durch den Wind hervorgerufene Verschiebung des Air-Frames gegen den Earth-Frame seit Beginn des Kreisflugs und korrigiert die Abstandsmessung zwischen Flugzeug und Kreismittelpunkt entsprechend. Als Stellgröße berechnet der Circle-Fly-Regler einen Korrekturwert $\varphi_{\text{correction}}$ für den berechneten Rollwinkel φ_{\max} . Das Ausgangs-Limit des *Heading-Hold*-Reglers wird auf die Summe $\varphi_{\text{limit}} = \varphi_{\max} + \varphi_{\text{correction}}$ gesetzt.

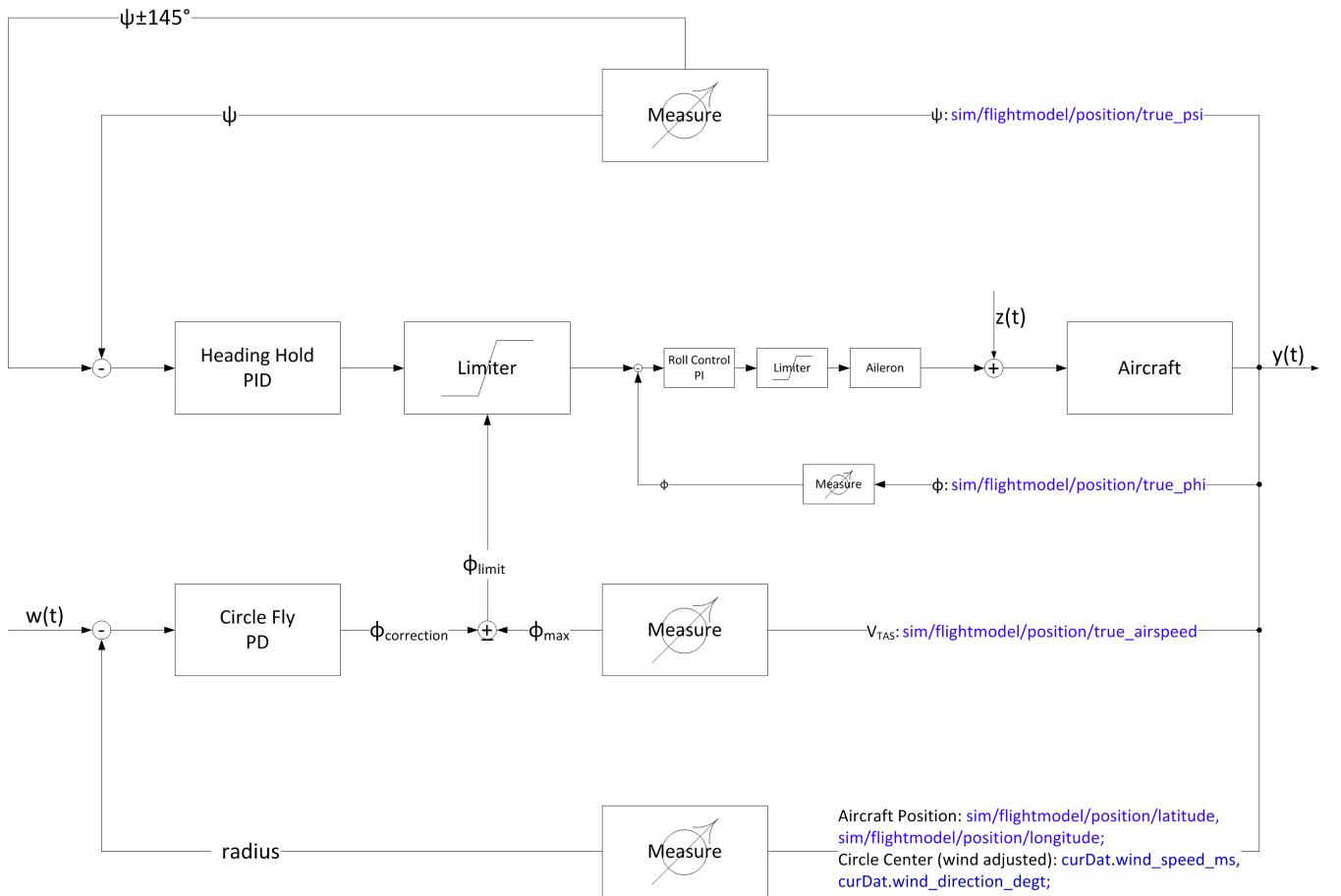


Abbildung 5.10: Reglerkaskadierung zum Steuern eines Kreisflugs.

Bei der Parametrierung des Reglers ergibt sich der Wert $K_I = 0$. Damit degeneriert der PID-Regler zu einem reinen PD-Regler.

Aufgrund der permanenten Übersteuerung des *Heading-Hold*-Reglers durch das dauernde Weiterdrehen der Zielrichtung läuft der Integrator während eines Kreisflugs auf sein Wind-Up-Limit und verbleibt dort. Wenn der Kreisflug beendet wird, muss daher auf jeden Fall der Integrator des *Heading-Hold*-Reglers zurückgesetzt werden, da es sonst viel zu lange dauert, den Integratorwert wieder abzubauen. Das würde zu einem massiven Überschwinger beim Übergang vom Kreisflug in den Geradeausflug führen.

ZUSAMMENSPIEL UND AUFRUF DER REGLER In der Beschreibung der einzelnen Regler wird deutlich, dass die komplexeren Funktionen durch eine Kaskadierung mehrerer Regelkreise realisiert werden. Übergeordnete Regelkreise berechnen Sollwerte, bzw. definieren Steuergrößenlimits für untergeordnete Reglerinstanzen. Letztendlich

steuern lediglich die beiden Regler für Sink-Control und für Roll-Control das Flugzeug über das Höhen- und Querruder. Alle anderen vorgestellten Funktionen werden durch Modifikation der Sollwerte und Limits für diese beiden Regler erreicht.

In der Implementierung des Autopiloten werden alle Regelkreise timergesteuert alle 100 Millisekunden in einer Schleife aufgerufen. Die Stellwerte übergeordneter Regler werden direkt vom aufgerufenen Regler als Sollwerte für die untergeordneten Regler weiter gereicht. Trotz des deutlich unterschiedlichen Routings der einzelnen Signale kann dies weitgehend generisch erfolgen, da die Funktion zur Rückgabe der Stellwerte `publishOutput` in jedem Regler mit einem passenden Funktionsobjekt überschrieben wurde und damit ein einheitliches Interface unabhängig vom konkreten Signalrouting zur Verfügung steht.

Bei künftigen Erweiterungen ist darauf zu achten, dass die Reihenfolge der Regleraufrufe eingehalten wird: zunächst müssen die übergeordneten Regler die Sollwerte für untergeordnete Regler berechnen, bevor diese schlussendlich Stellwerte für das Flugzeug ermitteln.

5.3 Darstellung der Reglerwerte und interaktive Parametrierung

Im interaktiven Modus des Autopiloten können Sollwerte für die einzelnen Regler über die GUI vorgegeben werden. Die SW sorgt automatisch dafür, dass bei Aktivierung eines übergeordneten Reglers auch die benötigten tieferliegenden Regelkreise in der Kaskade aktiviert werden. Die manuelle Sollwertvorgabe ist in den nachgeordneten Reglern deaktiviert.

Die Soll- und Ist-Werte der Regler werden zur nachträglichen Auswertung in eine Logdatei geschrieben. Insbesondere bei der interaktiven Reglerparametrierung ist aber eine direkte Rückmeldung an den Benutzer notwendig. Daher wird im Dialogfeld zur Parametrierung der Regler in Echtzeit ein Graph des Stellwerts und des Ist-Werts angezeigt. Vertikal lassen sich die Graphen beliebig skalieren und verschieben,¹⁷ während auf der Zeit-Achse eine feste Skala gewählt wurde und immer die letzten 500ms angezeigt werden. Da immer nur der Graph eines Reglers sichtbar ist hilft diese feste Zeitskalierung dabei, Ereignisse und Wirkungen übergeordneter auf untergeordnete Regelkreise zu korrelieren.

Ein Beispiel ist in Abbildung 5.11 zu sehen in dem drei Zeitpunkte markiert sind: Zunächst befindet sich das Flugzeug im Geradeausflug. Als Gleitwinkel ist $\gamma_s = -5,0^\circ$ eingestellt. Es wird ein Kreisflug nach rechts mit einem Radius von 450m gestartet. Der Sollwert des Sink-Control Reglers wird auf einen Gleitwinkel von $\gamma_c = -5,5^\circ$ verändert. Die Dialogseiten für Roll-Control und Heading-Hold sind nicht gezeigt. Nach ca. 34 Sekunden wird der Kreisflug beendet, woraufhin der Gleitwinkel wieder auf

¹⁷ Mit der Maus lassen sich die Graphen vertikal verschieben und mit dem Rollrad vertikal skalieren. Aktionen ohne Modifier betreffen den Graphen des Ist-Werts, Aktionen mit gedrückter Ctrl-Taste den Graphen des Stellwerts.



Abbildung 5.11: Die Darstellung der Reglerwerte und Reglerparameter während eines Kreisflugs.

den Wert für den Geradeausflug zurückgesetzt wird. Kurz danach wird der Simulator pausiert, um Screenshots aufzunehmen. Die Werte rechts davon sind als ungültig zu betrachten.

In den Graphen der Regler ist jeweils in grün der Ist-Wert und in blau der Sollwert eingetragen. Der aktuelle Sollwert wird durch eine horizontale schwarze Linie gekennzeichnet, jedoch nicht über die Zeit aufgetragen.

In der Abbildung 5.11 sind auch die Schieberegler für die Reglerparameter K_p , K_I und K_d zu sehen. Mit diesen Schiebern kann im interaktiven Modus jeder einzelne Reglerparameter in Echtzeit verändert werden. Die Auswirkungen auf die Regelung

können direkt im darunter liegenden Graphen der Reglerwerte beobachtet und beurteilt werden. Um Sollwerte vorzugeben und während der Reglerparametrierung Sollwertsprünge auf das System zu geben kann der Drehregler oben rechts im Reglerdialog benutzt werden. Dieser Drehregler ist nur wirksam, wenn kein übergeordneter Regler aktiviert ist und so die Sollwerte vorgibt.

Die Parameter müssen für jeden Flugzeugtyp ermittelt werden. In der vorliegenden Arbeit wurden nur Versuche mit der in X-Plane mitgelieferten Cessna 172 durchgeführt. Alle im Text genannten Parameter beziehen sich daher auf dieses Flugzeugmodell.

5.4 Implementierung der Ablaufsteuerung

Eine vorberechnete Bahn besteht immer aus vier Bahnprimitiven: LSLS oder RSRS. Oben wurde beschrieben, wie mittels der Localizer-Interception und dem Heading-Hold eine gerade Teilstrecke abgeflogen werden kann und wie mittels des Circle-Fly-Reglers ein Kreissegment im Air-Frame realisiert wird.

Zur vollständigen Verfolgung eines vorberechneten Pfades fehlt noch eine Komponente zur Ablaufsteuerung. Sie muss automatisch die einzelnen Regler nacheinander mit den richtigen Parametern aktivieren und so die einzelnen Segmente zu einem Gesamtflug zusammensetzen. Diese Komponente wird in diesem Abschnitt beschrieben.

In der Ablaufsteuerung wird ein Bahnprimitiv als sogenanntes `FlightPhase`-Objekt repräsentiert, welches lediglich zwei Methoden unterstützt. Mittels `performFlight()` kann es dazu aufgefordert werden das Flugzeug entlang des spezifischen Bahnprimitivs zu steuern, indem die entsprechenden Regler aktiviert und mit passenden Sollwerten versorgt werden. Darüber hinaus kann über die Methode `isFinished()` abgefragt werden, ob der Flug entlang des aktuellen Bahnsegments schon beendet ist.

Das korrekte Aneinanderreihen der einzelnen Flugsegmente übernimmt ein Objekt namens `DubinsScheduler`. Dieses initialisiert eine Liste mit den einzelnen aufeinander folgenden Bahnprimitiven und ruft auf den zugehörigen `FlightPhase` Objekten timergesteuert wiederholt die beiden Methoden `performFlight()` und `isFinished()` auf. Sobald eine Flugphase als beendet gemeldet wird, iteriert der Scheduler seine Liste, um das nächste Segment zu aktivieren.

Es gibt drei `FlightPhase`-Spezialisierungen, die im Folgenden kurz beschrieben werden sollen:

- `StraightPhase` für gerade Segmente S
- `CirclePhase` für Kreissegmente entweder L oder R
- `RunOutPhase` als künstliches Abschlusssegment

STRAIGHTPHASE Ein gerades Segment ist charakterisiert durch seine Start- und seine Endposition. Damit ist auch die gewünschte Richtung ψ_{target} definiert. Um das

Flugzeug auf die Verbindungsgeraden zwischen Start und Ende, den Localizer für dieses Segment, zu steuern wird der einfacher P-Regler zur Localizer Interception benutzt. Aufgrund seines sehr einfachen Aufbaus ist dieser direkt im StraightPhase Objekt implementiert und gibt bei Aufruf der `performFlight()`-Methode einen aktualisierten Sollwert für den Heading-Hold Regler aus.

Da bei kürzer werdenden Distanzen zum Endpunkt die Winkelabweichungen bei noch bestehendem Abstand zur Localizer-Linie die in den P-Regler eingehen immer größer, und damit die Ausgleichsbewegungen immer heftiger werden, wird bei Unterschreiten einer bestimmten Distanz zum Zielpunkt auf ein einfaches Heading-Hold mit Sollwert ψ_{target} umgeschaltet. Das Flugzeug fliegt dadurch zwar ein paar Meter am Zielpunkt vorbei, hat aber die korrekte Richtung zum tangentialen Übergang in den Kreisflug. Es hat sich als wichtiger herausgestellt beim Übergang den richtigen Winkel zum Kreismittelpunkt einzuhalten als den genauen Radius. Dieser wird im nachfolgenden Kreissegment ja über die Circle-Fly-Regelung korrigiert.

Das tangentiale Berühren des folgenden Kreissegments wird daher auch als Endkriterium für die StraightPhase verwendet. Der Geradeausflug wird beendet, wenn der Winkel zwischen der aktuellen Flugrichtung ψ und der Verbindung zwischen aktueller Position und nächstem Kreismittelpunkt 90° beträgt, der Kreis also tangential berührt wird. Um ein einheitliches Abbruchkriterium zu haben, wird für das letzte S-Segment, dem kein Kreisflug mehr folgt, bei der Initialisierung ein künstlicher Kreismittelpunkt erzeugt der links oder rechts des Endpunkts liegt.

CIRCLEPHASE Ein Kreissegment ist durch seinen Mittelpunkt und seinen Start- und Endwinkel charakterisiert. Es muss lediglich der Circle-Fly Regler für einen Kreisflug aktiviert und der geflogene Gesamtwinkel auf integriert werden. Zur Erkennung des Segmentendes wird verglichen ob der gesamte erforderliche Kreiswinkel schon erreicht wurde. Dabei ist zu beachten, dass ein Kreissegment auch einen Winkel von mehr als 360° überstreichen kann. Empirisch wurde ermittelt, dass der Kreisflug schon bei einem Winkel 5° vor dem gewünschten Endwinkel beendet werden sollte. Das verhindert ein Überschwingen am Kreisende deutlich, das sonst auftreten würde, da der Übergang vom Kreis- in den Geradeausflug eine gewisse Zeit braucht.

RUNOUTPHASE Als künstliches letztes Segment wird in den DubinsSchedule eine RunOut-Flugphase eingefügt. Diese dient lediglich dazu, alle aktiven Regler zu deaktivieren und das Flugzeug in eine stabile Fluglage zu setzen, falls es noch nicht auf dem gewünschten Landeplatz aufgesetzt hat. Das passiert regelmäßig bei Flugversuchen mit virtuellen ELF. In der RunOut-Phase werden auch die während der vorhergehenden Flugphasen erfassten Messdaten in einen einzigen Datensatz zusammengefasst.

5.5 Berücksichtigung des Windes und Erfassung von Messdaten

Der implementierte Autopilot arbeitet im Air-Frame, der seine Position gegenüber dem Earth-Frame verändert. Aus praktischen Gründen (z. B. zur Visualisierung) werden die vorberechneten charakteristischen Bahnpunkte (Kreismittelpunkte, Anfangs- und Endpunkte der geraden Strecken) jedoch in WGS84-Koordinaten, also im Earth-Frame gespeichert.

Beim Abfliegen einer Bahn müssen diese Punkte wieder korrekt in den Air-Frame transformiert werden. Dazu wird der von X-Plane ausgegebene Windgeschwindigkeitsvektor¹⁸ kontinuierlich aufintegriert. Zu Beginn einer Bahnverfolgung wird ein Nullpunkt gesetzt. Die Transformation der Earth-Frame Koordinaten in den Air-Frame erfolgt davon ausgehend durch einfache Vektoraddition des Integrals zu den Koordinaten der charakteristischen Punkte.

Diese Windkorrektur ist beim Abfliegen der Segmenten im Dubins-Pfad notwendig. Sie wird daher in den Reglern für Circle-Fly und Localizer-Interception angewendet.

Auch die während der Flugregelung erfassten Messdaten müssen um den Windeinfluss korrigiert werden. In jedem Timerschritt wird der zurückgelegte Weg des Flugzeugs erfasst. Da die von X-Plane gelieferten Positions-Angaben in WGS84-Koordinaten vorliegen müssen diese zunächst in den kartesischen Earth-Frame und anschließend noch, um den Windeinfluss seit dem letzten Messwert korrigiert, in den Air-Frame umgerechnet werden. Mit der Erfassung der Wegdaten im Air-Frame kann nach Abschluss des Anflugs ein Vergleich der tatsächlichen Flugbahn mit der im Air-Frame geplanten Bahn gezogen werden.

Um eine visuelle Kontrolle der abgeflogenen Bahnen auch unter Windeinfluss zu ermöglichen, wird die geplante Dubins-Kurve nach Abschluss des Anflugs in den Earth-Frame transformiert und in der Visualisierung eingeblendet. Dazu werden die Windintegrale verwendet, die während des Anflugs aufgenommen wurden. Ein Beispiel dieser nachträglichen Transformation ist in Abbildung 6.9 gezeigt.

Neben den beschriebenen Komponenten, die eng mit der Domänenlogik des Autopiloten und der Bahnanalyse verzahnt sind, sollen noch kurz zwei weitere, eher technische Komponenten angesprochen werden, da sie für das Evaluationssystem wichtig sind.

¹⁸ X-Plane hat bei der Angabe des Windes einen Bug, der zu berücksichtigen ist: Der Wert, der in der Dataref `sim/weather/wind_speed_kt` von X-Plane ausgegeben wird ist in $\frac{m}{s}$ angegeben und nicht in Knoten. Wenn man den Wind dagegen über die Datarefs `sim/weather/wind_speed_kt[]` schreibt, so erwartet X-Plane einen Wert in Knoten.

5.6 Datenverbindung zum Simulator

Um die Experimente mit dem implementierten Autopiloten durchzuführen, wird die kommerzielle Flugsimulations-Software *X-Plane* verwendet. Sie wird von der Firma Laminar Research entwickelt und vertrieben.

Der Simulator X-Plane lässt sich weitgehend durch externe Programme fernsteuern. Dazu bietet er sowohl eine Plugin-Schnittstelle über die sich externe Programme direkt einbetten lassen, als auch eine UDP-Netzwerkschnittstelle über die zyklisch interne Werte ausgegeben und Befehle entgegengenommen werden können.

In dieser Arbeit fiel die Wahl der Schnittstelle auf das UDP-Interface, da dies eine sehr lose Kopplung zwischen selbst entwickelter Autopilot-Software und der Simulationsplattform ermöglicht. Der Autopilot abonniert eine Reihe von Messwerten und setzt mit einer Update-Rate von 10 Hertz Eingabebefehle für das Steuerhorn ab. Durch den Autopiloten wird das Höhenruder (Elevator) und das Querruder (Aileron) kontrolliert. Eine Kontrolle des Seitenruders (Rudder) findet aktuell nicht statt.

Durch die Beschränkung auf ein solch einfaches Interface ist die entwickelte Software grundsätzlich darauf vorbereitet auch in anderen Simulationsumgebungen oder sogar realen Flugversuchen eingesetzt zu werden. Dazu muss lediglich ein einfacher Wrapper entwickelt werden, der die Messwerte regelmäßig bereitstellt und die Steuerbefehle an Aktuatoren ausgibt. Für die Flugregelung werden keine speziellen Funktionen benutzt, welche nur in X-Plane zur Verfügung stehen.

X-Plane bietet in der aktuellen Version mehrere Tausend sogenannte *DataRefs* an, die interne Zustände des Simulationsmodells oder z. B. von Eingabegeräten oder Instrumenten zugänglich machen.¹⁹ Sie werden über das Netzwerkprotokoll UDP zugänglich gemacht. Viele dieser DataRefs sind nicht nur lesbar, sondern auch beschreibbar, so dass sich der interne Zustand des Simulators verändert lässt.

Die DataRefs lassen sich bei X-Plane abonnieren und werden dann mit der gewünschten Update-Rate per UDP aktualisiert. Da die verschiedenen DataRefs unterschiedliche maximale Update-Raten haben, und diese auch noch ziemlich stark jittern, ist bei der Verwendung der Daten darauf zu achten eigene Zeitstempel einzufügen. Im Allgemeinen liegt die maximale Update-Rate bei 20Hz. Eine Liste der vom Autopiloten verwendeten DataRefs findet sich im Anhang A.3.

¹⁹ Eine Liste aller DataRefs findet man im Installationsverzeichnis von X-Plane in der Datei `./Resources/plugins/DataRefs.txt`. Eine besser zugängliche und durchsuchbare Liste bietet die Webseite <https://www.siminnovations.com/xplane/dataref/index.php>. Die Dokumentation der DataRefs beschränkt sich in der Regel lediglich auf einen sehr kurzen Kommentar zusammen mit der Angabe, ob der Wert nur lesbar, oder auch beschreibbar ist. Mit etwas Phantasie und Google lässt sich trotz der sehr knappen Dokumentation für die meisten Zwecke die korrekte DataRef finden.

5.7 Datenhaltung und Datenausgabe

Zur Datenhaltung innerhalb der Autopilot Software wurde ein Objekt `DataCenter` erstellt. Dieses Objekt abonniert die gewünschten DataRefs bei X-Plane, kümmert sich um die komplette Datenhaltung und das Logging und stellt allen anderen Programmteilen den Zugriff auf die jeweils letzten Zustandsdaten bereit. Das `DataCenter` ist als Singleton-Objekt realisiert, so dass alle Programmteile Zugriff auf die gleichen, konsistenten Daten erhalten, ohne dass eine Objektreferenz durch das Objektgeflecht gereicht werden muss.

Die zentrale Datenhaltung im `DataCenter` hat den Vorteil, dass beim Einsatz der Autopilot-Software in anderen Umgebungen lediglich die Datenquelle dieses einen Objekts geändert werden muss, um z. B. anstelle von Simulationsdaten per UDP, reale Flugdaten von Sensoren eines realen Flugzeugs zu verwenden. Das Interface zu den anderen Programmteilen bliebe unverändert.

Für die intensive, nachträgliche Auswertung der Flugversuche wurde eine Logging-Funktion implementiert. Je nach Einstellung und Benutzerwunsch werden alle von X-Plane abonnierten Daten sowie eine Vielzahl von im Autopiloten berechneter Werte kontinuierlich in eine Textdatei geschrieben. Als Datenformat wurde das einfache und menschenlesbare CSV-Format (comma separated values) gewählt, das mit jeder üblichen Tabellenkalkulation weiter analysiert werden kann.

Da das Data-Logging auch automatisiert gestartet werden kann, wird automatisch ein Dateiname mit einem Zeitstempel im Namen erzeugt. Dadurch lassen sich die Log-Dateien im Anschluss leicht zuordnen.

Neben diesen vollständigen Detaildaten im Zeitverlauf werden noch umfangreiche zusammengefasste Messdaten bezüglich der Güte der Bahnverfolgung errechnet. Diese werden im JSON-Format erzeugt, so dass sie bequem über die REST-API der Visualisierungskomponente zur Verfügung gestellt werden können. Ein Beispiel für die so erstellte Zusammenfassung eines vollständigen Anflugs ist in Abschnitt A.2 gezeigt.

Damit sind die wichtigsten Aspekte der Implementierung erklärt. Weitere Details finden sich im Anhang sowie unter <https://github.com/opt12/DubinsPilot> im Source-Code auf Github.

6 PARAMETRIERUNG UND EVALUATION DES AUTOPILOTEN

Zur Qualitätsbewertung der Flugregelung des implementierten Autopiloten sind zunächst einige grundlegende Tests durchzuführen. Dazu gehört zu Beginn die Ermittlung der Parameter für die Regelkreise und sinnvoller Werte für die Gleitwinkel im

Geradeaus- γ_s und im Kurvenflug γ_c zusammen mit dem zu benutzenden Kreisradius r .

Unter Verwendung der ermittelten Parameter wird danach die Bahnverfolgung überprüft. Das erfolgt sowohl im Windstillen, als auch mit konstantem Wind. Es wird bewusst ein sehr einfaches Kriterium zur Bewertung des Autopiloten herangezogen: Als relevante Daten werden in dieser Auswertung lediglich der Abstand vom vorgegebenen Zielpunkt sowie die Höhenabweichung am Ende der Bahnverfolgung verwendet.

Auf die Problematik, die sich bei der Voraussage der sich einstellenden Geschwindigkeit und der damit eng verbundenen Voraussage der Flugdauern ergibt, wird zusätzlich in der Interpretation der Ergebnisse eingegangen. Diese Voraussagen sind notwendig, um den Einfluss des Windes schon bei der Bahnplanung zu antizipieren und entsprechend korrigieren zu können. Wie schon in der Beschreibung des Algorithmus in Abschnitt 3.3 erwähnt ist diese Korrektur selber jedoch außerhalb des Fokus dieser Arbeit.

6.1 Ermittlung der Grundparameter

Die ersten Experimente zur Evaluation der Flugregelung konnte schon recht früh im Projektverlauf erfolgen. Dazu mussten zunächst nur die Regler für Sink-Control und für Roll-Control implementiert werden. Mit Implementierung der weiteren Regler konnten nach und nach weitergehende Flugversuche durchgeführt werden.

6.1.1 Ermittlung der Reglerparameter

Da die Regelstrecke vollständig als Black-Box angesehen wurde, mussten die Reglerparameter empirisch ermittelt werden. Dazu wurden die in der GUI eingebauten Schieberegler und die Kurven von Ist- und Stellwert verwendet. Begonnen wurde mit einer groben Einstellung in Anlehnung an das Verfahren von Ziegler und Nichols (nach [Berger, 2001, §9.3.1]) begonnen und die damit gewonnenen Anfangswerte im Verlauf der Entwicklung und Erprobung immer weiter verfeinert.

In Abbildung 6.1 ist zu sehen, wie zu Beginn der Parametrierung versucht wurde die kritische Verstärkung des Systems zu ermitteln, bei dem es zu einer Dauerschwingung kommt. Es ist deutlich zu sehen, dass die Regelstrecke für den Rollwinkel um Größenordnungen schneller reagiert als die für den Gleitwinkel. Daher ließ sich die Roll-Control wesentlich einfacher parametrieren als die Sink-Control. Während der Gleitwinkel bei kritischer Verstärkung dauerhaft schwingt geht der Rollwinkel bei minimalen Änderungen des Verstärkungsfaktors sehr schnell wieder in einen stabilen Zustand über.

Aus der Erfahrung und mit den Angaben aus Tabelle 5.1 ließen sich die Veränderungen immer besser abschätzen, so dass inzwischen ein Satz stabiler Reglerparameter für die Flugversuche mit der in X-Plane modellierten Cessna 172 entstanden ist. Die-

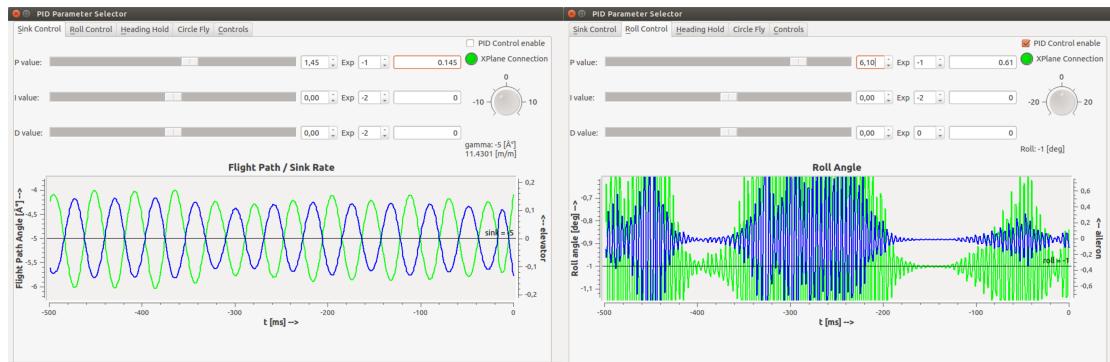


Abbildung 6.1: Interaktive Einstellung der Reglerparameter für Sink- und Roll-Control. Zu sehen ist das Schwingen der Regler bei kritischer Verstärkung K_p .

se Parameter sind in Tabelle 6.1 angegeben. Mit diesem Parametersatz wurden alle weiteren Experimente durchgeführt.

Parameter	Sink-Control	Roll-Control	Heading-Hold	Circle-Fly
K_p	0.09	0.09	0.65	0.15
K_I	0.07	0.03	0.07	0.00
K_d	0.07	0.00	0.07	0.20

Tabelle 6.1: Interaktiv ermittelte Reglerparameter für die Regelkreise des Autopiloten.

Wenn in Zukunft Versuche mit anderen (simulierten) Flugzeugtypen durchgeführt werden sollen, so müssen diese Parameter für jeden Flugzeugtyp individuell ermittelt werden.

6.1.2 Gleitwinkel und Geschwindigkeiten

Es ist wünschenswert in einer Notsituation noch einen möglichst großen Aktionsradius zur Verfügung zu haben. Daher soll für den Gleitwinkel ein Wert möglichst nah am „best glide“ verwendet werden. Dabei ist darauf zu achten, dass noch ausreichend Regelreserve zur Verfügung steht, da im realen Flug Störungen sowohl nach oben als auch nach unten vorkommen und ausgeregelt werden müssen. Der Winkel γ für den best glide Fall ist durch die Tangente an die Gleitpolare (vgl. Abbildung 2.2) definiert. Dies ist der kleinste erreichbare Winkel, um die Gleitpolare gerade noch zu berühren. Beim Flug im best glide ist es daher nicht möglich zu flacheren Winkeln hin auszuregeln, weil dann die Gleitpolare verlassen wird. Falls ein zu flacher Gleitwinkel gewählt wird, muss das Höhenruder immer weiter gezogen werden, bis es schließlich zum Strömungsabriss kommt. Die Folgen eines solchen Überziehens sind in Abbildung 6.2 zu sehen. Es muss daher ein Gleitwinkel gewählt werden, der zwar nahe am „best glide“

Winkel liegt, jedoch etwas steiler ist, damit die Regelung zuverlässig funktionieren kann.

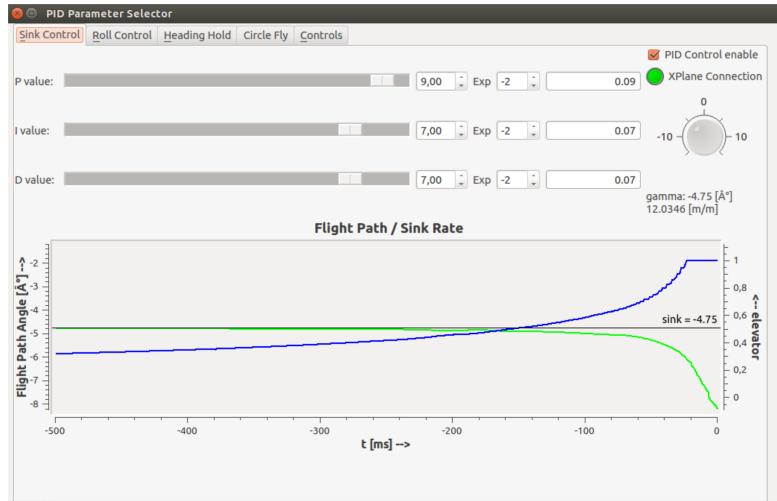


Abbildung 6.2: Bei einem zu regelnden Gleitwinkel von $\gamma_s = 4,75^\circ$ muss das Höhenruder immer weiter gezogen werden, bis es schließlich zum Abriss kommt. Der Gleitwinkel war zu flach gewählt.

Um die zu verwendenden Gleitwinkel zu ermitteln wurden viele Flugversuche durchgeführt, in denen die Winkel immer flacher gewählt wurden, bis sich herausstellte, dass das Flugzeug den gewählten Winkel nicht mehr dauerhaft halten kann. Es stellte sich heraus, dass in größerer Höhe flachere Winkel möglich sind als in niedrigeren Luftsichten. Die Konstanz des best glide ist offenbar nur eine erste Näherung und nicht vollständig gültig.

Für die verwendete Cessna 172 wurde als Wert sehr nahe am best glide Winkel $\gamma_{\text{best}} \approx -4,9^\circ$ ermittelt. Damit wäre jedoch keine Regelreserve mehr vorhanden. Deswegen wird als guter, aber noch ausreichend regelbarer Wert für den einzustellenden Gleitwinkel im Geradeausflug $\gamma_s = -5,0^\circ$ verwendet. In Abbildung 6.3 ist die Aufzeichnung eines Sinkflugs aus über 3000m Höhe bis nahe Null zu sehen. Man sieht, dass in allen Höhen der gewählte Gleitwinkel gehalten werden kann.

Bei der Ermittlung der Gleitwinkel wurden auch die Geschwindigkeiten IAS und TAS aufgezeichnet. In Abbildung 6.3 ist zu sehen, wie die TAS mit abnehmender Höhe immer weiter abnimmt und sich in Bodennähe dem Wert der IAS annähert. Die im Information Manual einer Cessna 172S [Cessna Aircraft Company, 2004] angegebene IAS für den best glide ist 68 KIAS. Das entspricht $34,98 \frac{\text{m}}{\text{s}}$ und liegt damit geringfügig unterhalb der IAS, die sich beim ermittelten optimalen Gleitwinkel von $\gamma_s = 4,9^\circ$ ergeben hat.

In der Abbildung ist auch gut zu erkennen, wie die Sinkrate in $[\frac{\text{m}}{\text{s}}]$ mit abnehmender Höhe genau wie die TAS betragsmäßig abnehmen, ihr Verhältnis, die Gleitzahl in

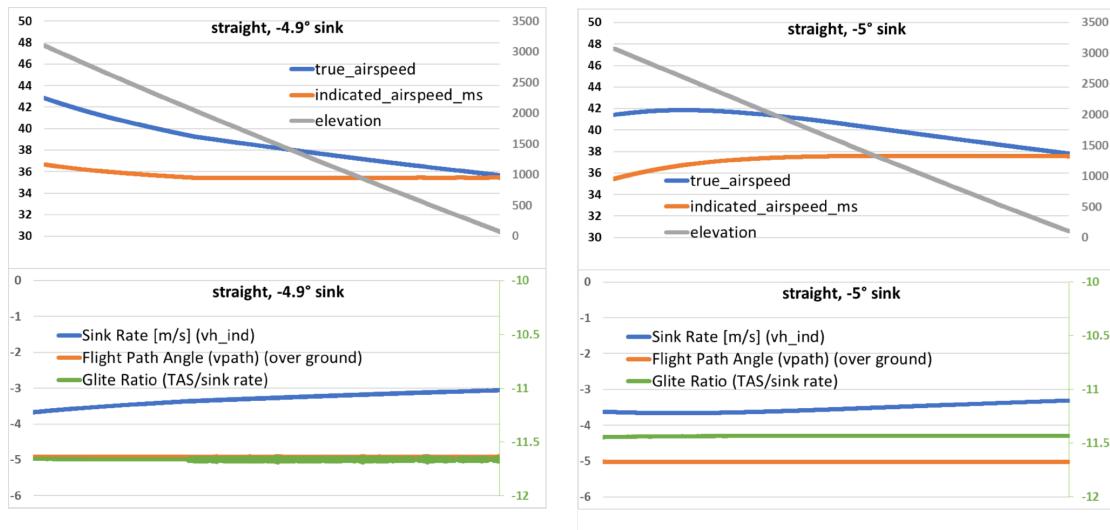


Abbildung 6.3: Aufzeichnungen von IAS, TAS, Flughöhe, Gleitwinkel, Sinkrate und Gleitzahl mit dem als optimal ermittelten Gleitwinkel von $\gamma_s = -4,9^\circ$ (links) und dem für den Autopiloten verwendeten Gelitwinkel von $\gamma_s = -5,0^\circ$ (rechts).

[$\frac{m}{s}$] und damit der Gleitwinkel in [$^\circ$] dabei aber konstant bleiben. Das bestätigt die in Abschnitt 2.3 geäusserte Feststellung, dass ein Flugzeug im best glide in größerer Höhe zwar schneller fliegt, dafür aber auch schneller sinkt.

6.1.3 Mögliche Kurvenradien

Bei der Ermittlung eines zu verwendenden Kurvenradius ist zu berücksichtigen, dass die Geschwindigkeit TAS mit der Höhe abnimmt. Da der Kurvenradius über den Rollwinkel quadratisch von der TAS abhängt (siehe Gleichung (5.11)), ist zu erwarten, dass bei geringerer Höhe und konstantem Kreisradius r der benötigte Rollwinkel φ abnimmt.

Das ist deutlich in Abbildung 6.4 zu erkennen. Hier ist der Sinkflug eines Flugzeugs bei konstantem Rollwinkel aufgezeichnet. Aufgrund der abnehmenden TAS in geringerer Höhe nimmt auch der Kreisradius stetig ab und es ergibt sich eine enger werdende Spirale.

In Abbildung 6.5 ist ein Kreisflug mit konstantem Radius zu sehen. Im Teildiagramm oben rechts sind der durch die Circle-Fly Regelung eingestellte Rollwinkel und der durch den Sink-Control Regler eingestellte Höhenruderausschlag um einen konstanten Gleitwinkel zu halten über der Höhe aufgetragen. In der Tendenz nimmt der benötigte Rollwinkel mit abnehmender Höhe ab. Eine genaue Voraussage ist aufgrund der Regleraktivität und der Störungen beim Ein- und Ausflug augenscheinlich nicht möglich.

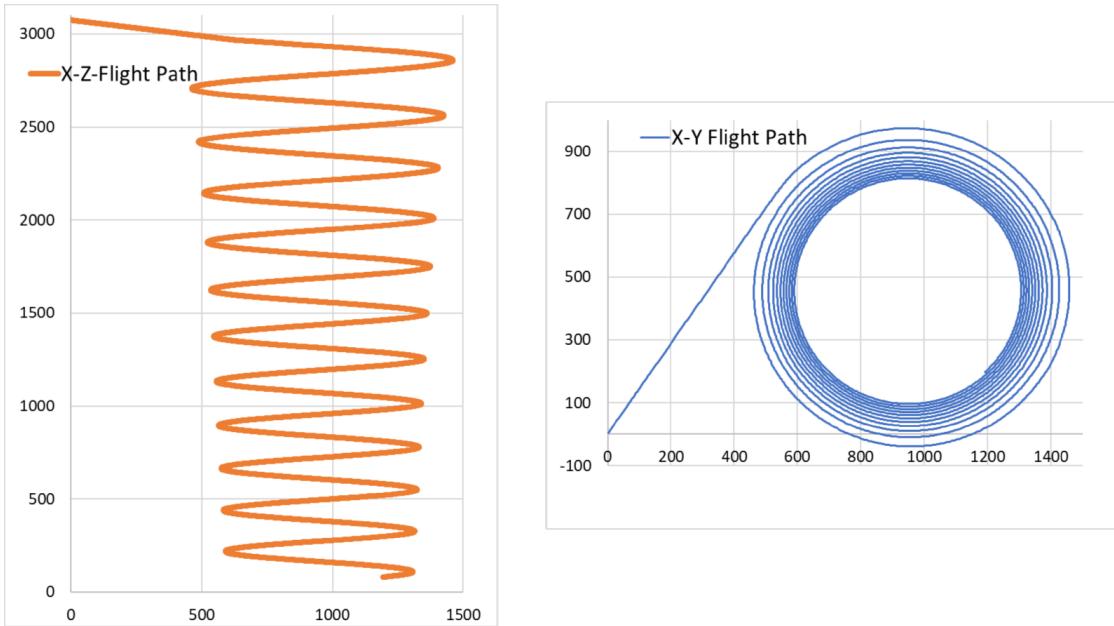


Abbildung 6.4: Sinkflug bei konstantem Rollwinkel von 25° . Zu sehen ist links die (x, z) -Position und rechts die (x, y) Position im Zeitverlauf.

Diese Änderung des benötigten Rollwinkels hat eine Verschiebung der optimalen Gleitpolare zur Folge. Offensichtlich kann nicht über die gesamte Höhe im best glide geflogen werden, wenn nur ein einziger Winkel γ_c zugrunde gelegt wird.

Während der Entwicklung wurde damit experimentiert den Gleitwinkel mit dem Rollwinkel nachzuregeln so dass gilt $\gamma_c = \gamma_c(\varphi)$, es hat sich aber gezeigt, dass diese Regelung sehr leicht instabil wird. Zudem ist für den verwendeten Bahnplanungsalgorithmus die Annahme eines bekannten Gleitwinkels im Kreisflug bei konstantem Radius unerlässlich.

Es muss daher ein Kurvenradius r und ein Gleitwinkel γ_c gewählt werden, der in allen vorkommenden Höhen erreicht werden kann. In vielen Flugversuchen wurden vernünftige Werte zu $r = 450\text{m}$ und $\gamma_s = -5,5^\circ$ ermittelt. In der Abbildung 6.5 ist die Aufzeichnung von IAS, TAS, Flughöhe, Gleitwinkel, Sinkrate und Gleitzahl dafür zu sehen.

ERMITTELTE PARAMETER IM ÜBERBLICK Für die weiteren Experimente wurden diese im Flugversuch ermittelten Flugparameter verwendet:

$$\begin{aligned}\gamma_s &= -5,0^\circ \\ \gamma_c &= -5,5^\circ \\ r &= 450\text{m}\end{aligned}\tag{6.1}$$

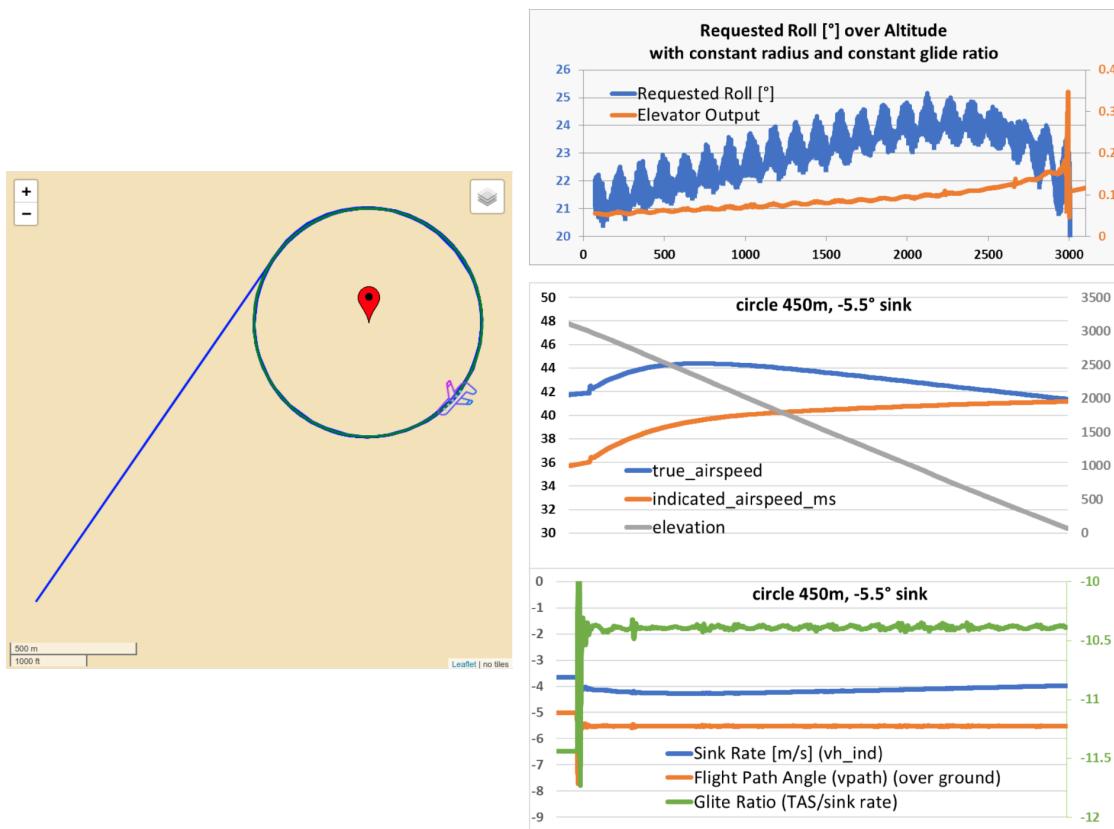


Abbildung 6.5: Oben rechts: Rollwinkel und Höhenruderausschlag für Kreisflug mit konstantem Radius $r = 450\text{m}$ und konstantem Gleitwinkel $\gamma_s = -5,5^\circ$ aufgetragen über der Höhe.
Darunter: Aufzeichnungen von IAS, TAS, Flughöhe, Gleitwinkel, Sinkrate und Gleitzahl mit dem als praktikabel ermittelten Gleitwinkel von $\gamma_s = -5,5^\circ$ aufgetragen über der Flugzeit.

6.2 Pfadverfolgung: Abfliegen geplanter Bahnen

Nach der Messung der Grundparameter und der Verifikation der grundlegenden Flugregelung (Sink-Control, Heading-Hold, Circle-Fly) soll überprüft werden, wie sich die Bahnverfolgung im Flugversuch bewährt.

Dazu wurden Flugversuche aus verschiedenen Höhen mit „virtuellen Landebahnen“ durchgeführt. Eine virtuelle Landebahn wird relativ zu den aktuellen Flugzeugkoordinaten angegeben, so dass die zu berechnende Flugbahn immer die gleiche Form annimmt, unabhängig von der Positionierung des Flugzeugs in Bezug auf die Erdoberfläche.

Die Position der virtuellen Bahn wird mit den beiden Koordinaten *forward* und *right* in [m], und mit der Ausrichtung *rotation* relativ zur aktuellen Flugrichtung in [°] an-

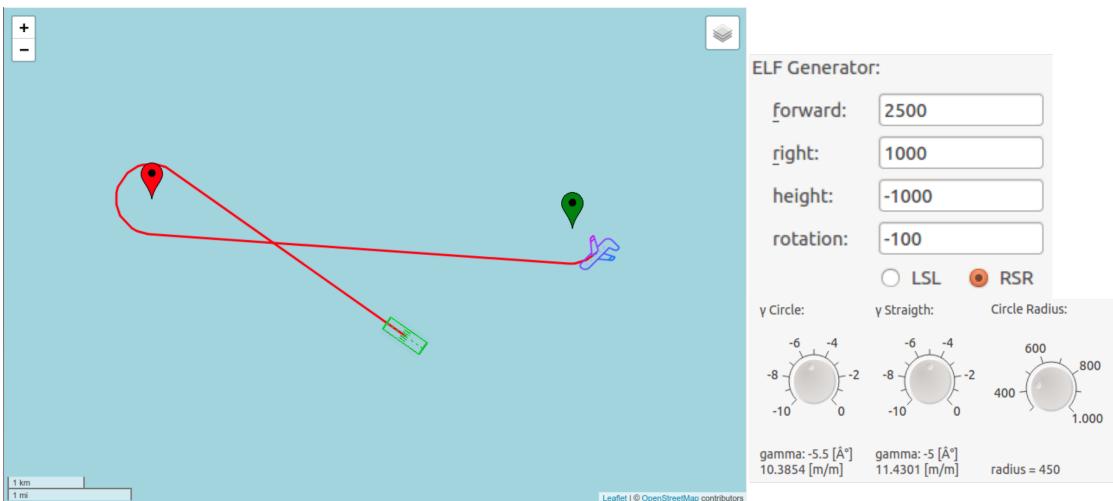


Abbildung 6.6: Berechnung und Parametrierung einer virtuellen Landebahn und der dorthin zu planenden Flugbahn.

gegeben. Die Höhe kann entweder relativ zur aktuellen Flughöhe oder aber in Metern über NN angegeben werden *height*. Außerdem kann die Berechnung des kürzesten möglichen Anflugs ohne Verlängerung des Dubins-Pfades gewählt werden. Die minimal mögliche Höhendifferenz wird dann automatisch ermittelt. Eine solche virtuelle Landebahn schwebt in den meisten Fällen „irgendwo in der Luft“, ohne dass das Flugzeug tatsächlich am Ende des Anflugs den Boden erreicht. Neben diesen Positionsparametern der virtuellen Bahn gehen in die Berechnung des Anflugs natürlich auch die ebenfalls einstellbaren Parameter für γ_s , γ_c und r ein. Eine virtuelle Landebahn ist zusammen mit ihren Parametern in Abbildung 6.6 dargestellt.

Obwohl das Szenario der virtuellen Landebahn nicht sehr realistisch klingt, können damit einfach –und vor allem automatisiert– viele Flugversuche durchgeführt werden, deren Ergebnisse miteinander vergleichbar sind. Bei Angabe absoluter Koordinaten der Zielbahn wäre eine Wiederholung identischer Anflüge mit sehr viel mehr Aufwand verbunden.

Zur Erprobung realer Anflüge auf reale Flugplätze ist auch die Angabe absoluter Koordinaten möglich.

6.2.1 Durchgeführte Experimente

FLIEGE IM WINDSTILLEN Zur Überprüfung der Grundlegenden Funktion der Bahnplanung und der Bahnverfolgung wurden zunächst Experimente im Windstilten durchgeführt. Dazu wurden zufällige virtuelle Bahnen mit verschiedenen Parametern definiert, die Anflüge dorthin berechnet und mit dem Autopiloten abgeflogen. Zur qualita-

tiven Beurteilung der Bahnverfolgung dient eine optische Kontrolle der vorausgeplanten Bahn und der während des Anflugs aufgezeichneten Spur der Flugzeugposition.

In Abbildung 6.7 sind zwei Beispiele von Anflügen zu sehen. Detailvergrößerungen sind in Abbildung 6.8 gezeigt. Damit lässt sich die Bahnverfolgung qualitativ beurteilen.

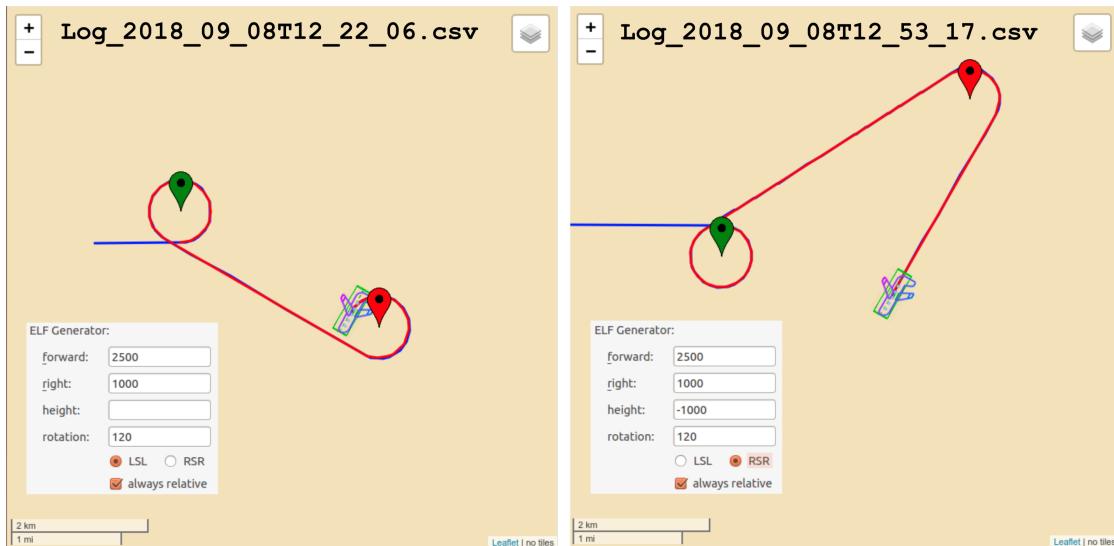


Abbildung 6.7: Zwei Anflüge, die mit dem Autopiloten durchgeführt wurden. Links ist ein unverlängerter Dubins-Pfad zugrunde gelegt, während rechts eine definierte Höhendifferenz zu überwinden ist.

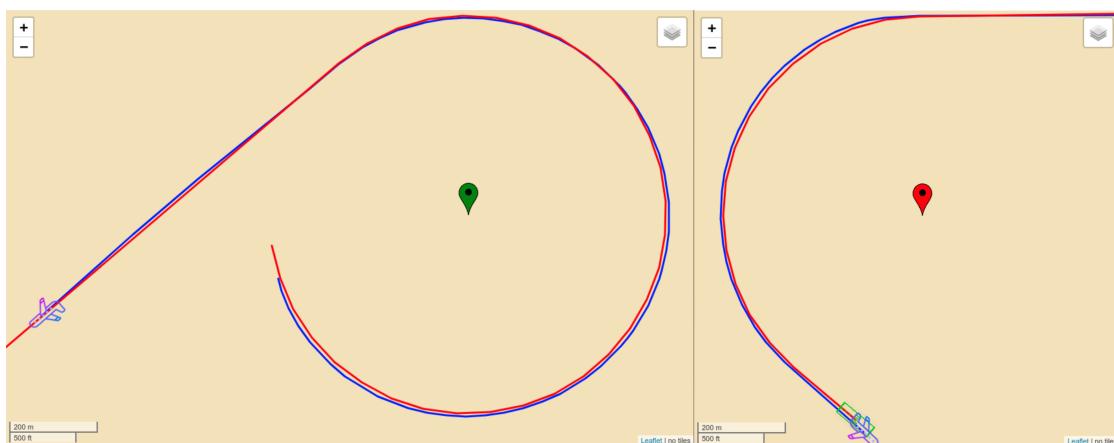


Abbildung 6.8: Detailvergrößerung zur Beurteilung der Bahnverfolgung.

Es wurden 22 Flugtests im Windstiller durchgeführt. Ein Auszug aus den Statistiken von 13 dieser Tests ist in Tabelle 6.2 zu sehen. Die Rohdaten, aus denen diese Tabelle extrahiert wurde, findet sich auf der beiliegenden CD im Verzeichnis

/FlightTestLogfiles/PathTrackingNoWind. Der Wert *distance* für den Abstand zum Zielpunkt wurde aus den Werten *x* und *y* berechnet.

FLIEGE MIT KONSTANTEM WIND Neben den Experimenten im Windstillen wurden auch Flüge mit konstantem Wind durchgeführt. Die Bahnplanung nimmt in der vorliegenden Implementierung noch keine Bahnkorrektur hinsichtlich des Windes vor. Daher ist zu erwarten, dass beim Abfliegen der Bahnen erhebliche Abweichungen zur im Earth-Frame visualisierten Bahn entstehen.

Dies gilt jedoch nur für die Form der Bahnen im dargestellten Earth-Frame. Da der Autopilot im Air-Frame arbeitet, sollten die geplanten Bahnen in Bezug auf dieses Bezugssystem trotz Wind getroffen werden.

Der Autopilot integriert kontinuierlich über den gemessenen Windvektor auf, so dass für jeden Bahnpunkt die Verschiebung zwischen Air-Frame und Earth-Frame bekannt ist. Diese Daten werden dazu benutzt nach Beendigung des Anflugs die im Air-Frame geplante Bahn korrekt in den Earth-Frame umzurechnen und zu visualisieren. Dazu wird die Verblasung durch den Wind jeweils für die Endpunkte der vier Segmente des Dubins-Pfades gespeichert und für die nachträgliche Transformation in den Earth-Frame benutzt. Die Kreissegmente werden dadurch zu Trochoiden verzerrt, während sich bei den geraden Segmenten nur Länge und Winkel verändern. Die so korrekt in den Earth-Frame transformierte Bahn wird nach Abschluss des Anflugs angezeigt, um auch hier eine qualitative Beurteilung der Bahnverfolgung vornehmen zu können.

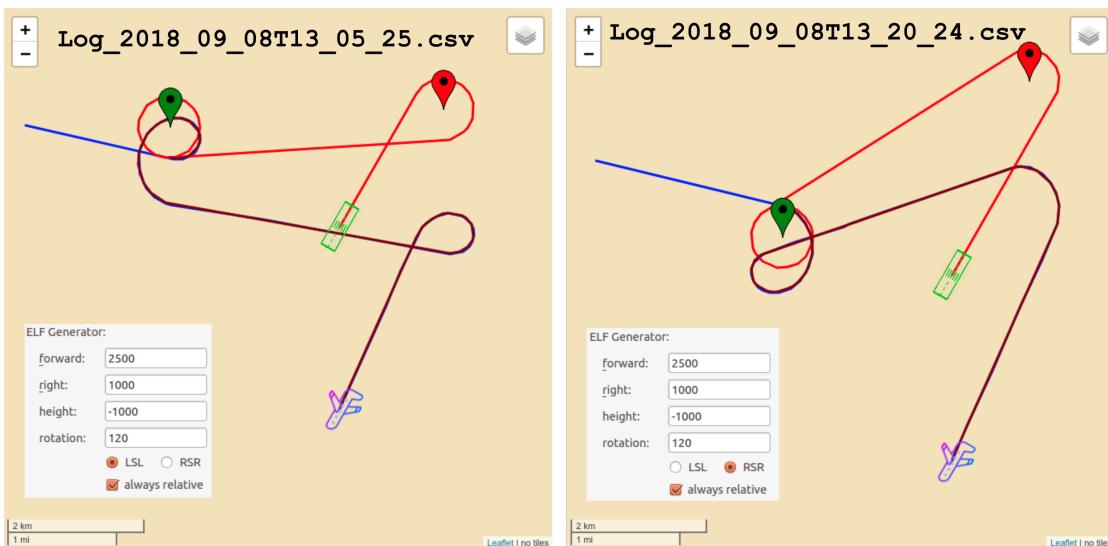


Abbildung 6.9: Anflüge mit konstantem Nordwind. Neben der untransformierten Bahn im Air-Frame ist die nachträglich in den Earth-Frame transformierte geplante Bahn eingezeichnet.

Zwei Beispiele von Anflügen in konstantem Wind sind in Abbildung 6.9 zu sehen. In beiden Fällen wurde konstanter Nordwind eingestellt, weil so die Verblasung sehr leicht zu interpretieren ist: Die aus dem Air-Frame zur Darstellung in den Earth-Frame transformierten charakteristischen Punkte liegen exakt südlich der entsprechenden Punkte in untransformierter Darstellung.

Insgesamt wurden 66 Flugtests mit konstantem Wind durchgeführt. Ein Auszug aus den Statistiken von insgesamt 13 dieser Tests ist in Tabelle 6.3 zu sehen. Bei diesen wurde konstanter Westwind eingestellt, damit die Verblasung nur in x -Richtung erfolgt und somit im Datensatz deutlicher zu sehen ist. Die Rohdaten, aus denen diese Tabelle extrahiert wurde, findet sich auf der beiliegenden CD im Verzeichnis /FlightTestLogfiles/PathTrackingWestWind. Der Wert $distance$ für den Abstand zum Zielpunkt im Air-Frame wurde aus den Werten x und y berechnet.

Tabelle 6.2: Auszug aus Flight-Test Statistiken für Flüge im Windstille (Wert *distance* berechnet)

plannedPath:													
pathLengthAF:	8624.29	10622.20	8280.03	5935.81	8827.52	5877.24	6523.75	9352.50	13640.25	10453.61	4038.94	10639.02	8274.20
rotation:	99.00	-91.00	-111.00	67.00	100.00	64.00	-131.00	137.00	57.00	161.00	27.00	-91.00	-111.00
heightLoss:	-787.00	-948.00	-766.58	-549.02	-789.00	-534.30	-612.43	-852.00	-1221.00	-949.82	-374.81	-948.00	-766.59
flownPath/airFrame/deviation:													
x	1.61	-3.56	-9.44	-9.20	-4.13	-2.88	-3.88	2.94	3.18	-2.19	-14.46	-3.23	-6.48
y	-5.83	-2.60	-7.07	8.93	0.73	-2.64	9.17	1.70	2.65	-2.40	-0.60	-0.56	5.51
distance	6.05	4.41	11.79	12.82	4.19	3.91	9.96	3.40	4.14	3.25	14.47	3.28	8.51
z	-3.47	-3.56	-3.94	-3.31	-4.51	-3.67	-3.63	-4.31	-4.36	-6.13	-4.71	-3.78	-2.53

Tabelle 6.3: Auszug aus Flight-Test Statistiken für Flüge mit konstantem Westwind (Wert *distance* berechnet)

plannedPath:													
pathLengthAF:	10642.50	8278.00	5932.94	8837.70	5888.57	6536.31	9350.71	13667.61	10463.05	4051.08	12004.75	12533.97	4659.87
rotation:	269.00	609.00	427.00	-260.00	-296.00	589.00	497.00	417.00	-559.00	-333.00	-186.00	-436.00	-328.00
heightLoss:	-948.00	-766.59	-549.03	-789.00	-534.31	-612.43	-852.00	-1221.00	-950.00	-374.81	-1063.00	-1123.00	-429.09
flownPath/earthFrame/deviation:													
x	2627.43	2003.73	1444.90	2168.16	1451.84	1572.79	2269.23	3369.39	2525.61	958.85	3019.95	3092.46	1138.35
y	-1.89	-0.01	12.65	1.05	1.02	7.64	-2.86	-3.49	-5.31	8.97	1.08	-0.18	-10.14
flownPath/airFrame/deviation:													
x	-4.56	-12.27	-1.10	-2.84	-5.15	3.80	2.23	5.40	-0.39	-24.15	0.95	0.47	-1.65
y	-0.49	0.58	13.06	1.71	1.55	8.84	-1.49	-0.37	-1.01	10.30	2.03	2.24	-8.85
distance	4.59	12.28	13.10	3.31	5.38	9.62	2.68	5.41	1.08	26.26	2.24	2.29	9.00
z	-4.35	-3.70	-3.80	-5.84	-5.01	-3.93	-4.08	-6.12	-6.01	-6.04	-3.83	-7.76	-4.57

6.3 Interpretation der Ergebnisse

ZIELERREICHUNG Für die Beurteilung des Autopiloten wird in dieser Auswertung lediglich die Genauigkeit der Zielerreichung betrachtet. Aus den während der Flugversuche gesammelten Daten werden die Angaben über den planaren Abstand zum Zielpunkt und der Höhenfehler am Ende des Anflugs extrahiert. Die Ergebnisse zusammen mit einigen Metadaten sind in den Tabellen 6.2 (windstill) und 6.3 (mit konstantem Wind) zu finden.

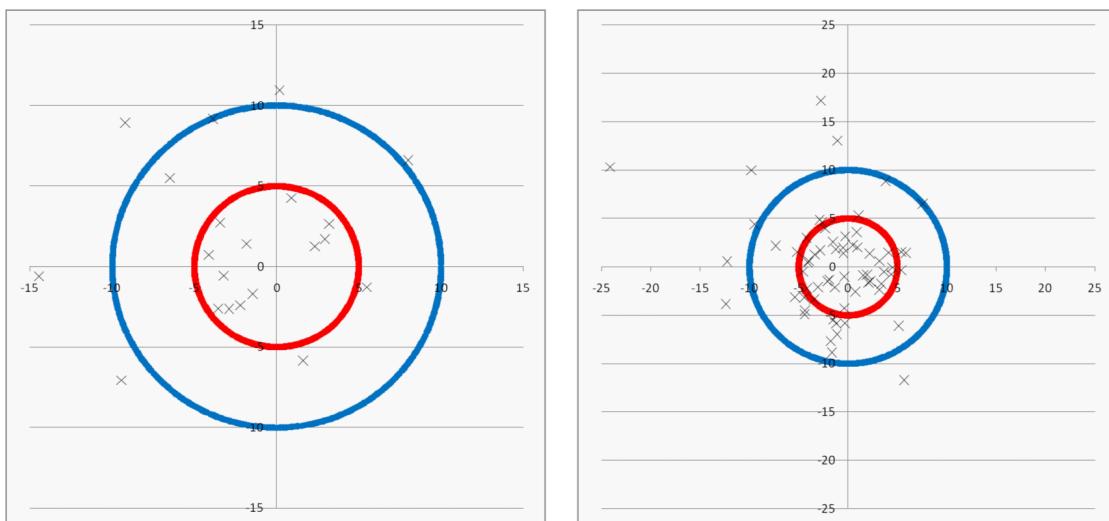


Abbildung 6.10: Abstände vom gewünschten Zielpunkt im Air-Frame am Ende des automatischen Anflugs über 22 Flugtests im Windstillen (links) und über 66 Flugtests mit $10 \frac{m}{s}$ Westwind (rechts) mit unterschiedlichen Parametern.

Erfreulich ist die Genauigkeit der Zielerreichung: In den allermeisten Fällen wird der gewünschte Zielpunkt mit einer Abweichung unter 10m erreicht. Der Abstand zum Ziel ist in Abbildung 6.10 als Scatter-Plot dargestellt. Dies erscheint für den intendierten Einsatz als völlig ausreichend. Wie nach Konstruktion erwartet, spielt der konstante Wind bei der Genauigkeit der Regelung keine relevante Rolle. Die Zielabweichungen bei den Flügen mit Wind sind bezogen auf den Air-Frame ähnlich denen im Windstillen.

HÖHENABWEICHUNG Die gewünschte Zielhöhe wird in allen Flugversuchen leicht unterschritten doch auch diese Abweichung ist erfreulich gering. Auch hier ergab sich kein signifikanter Unterschied zwischen den auftretenden Höhenfehlern im Fall mit Wind und im Fall ohne Wind. Das ist ein weiterer Hinweis darauf, dass die Berechnung der Flugbahnen im Air-Frame sinnvoll ist und die gewünschten Ergebnisse liefert. In Abbildung 6.11 wurde im oberen Diagramm die Abweichung über die gesamte

Fluglänge im Air-Frame aufgetragen. Es ergibt sich keine auffällige Abhängigkeit zwischen auftretendem Höhenfehler und dieser Fluglänge.

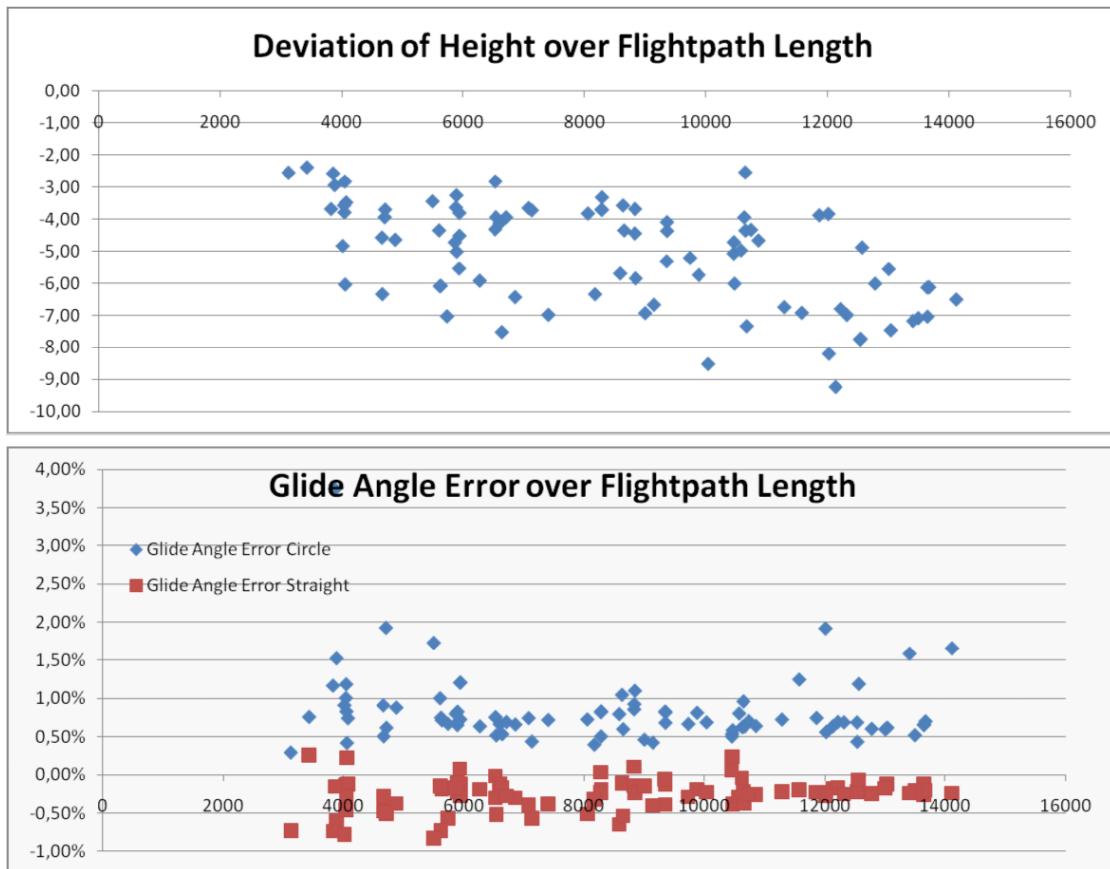


Abbildung 6.11: oberes Diagramm: Auftretender Höhenfehler über die gesamte Anfluglänge im Air-Frame
unteres Diagramm: Abweichung des erreichten Gleitwinkels γ_{ist} zum vorgewählten Gleitwinkel γ_{soll} für Kreis- und Geradeaus-Segmente

Eine nähere Betrachtung lässt vermuten, dass die Höhenabweichung hauptsächlich in den Kreissegmenten eingeflogen wird, während in den geraden Segmenten sogar minimal Höhe gut gemacht wird. Im unteren Diagramm von Abbildung 6.11 ist zu sehen, dass die Abweichung vom gewünschten Gleitwinkel in den Kreissegmenten erheblich größer ist, als im Geradeausflug. Im Geradeausflug ist eine leichte Tendenz zu erkennen, dass der erreichte Gleitwinkel eher flacher als steiler ist und somit die erreichte Höhe nach einem geraden Segment höher als erwartet ist.

Eine Betrachtung des Gleitwinkelplots aus dem Sink-Regler wie in Abbildung 6.12 legt die Vermutung nahe, dass diese Abweichung vor allem von den Wechseln zwischen Geradeaus- und Kurvenflug herrührt und damit unabhängig von der Länge der Segmente ist. An den Übergangsstellen von Geradeaus- zu Kurvenflug erfährt

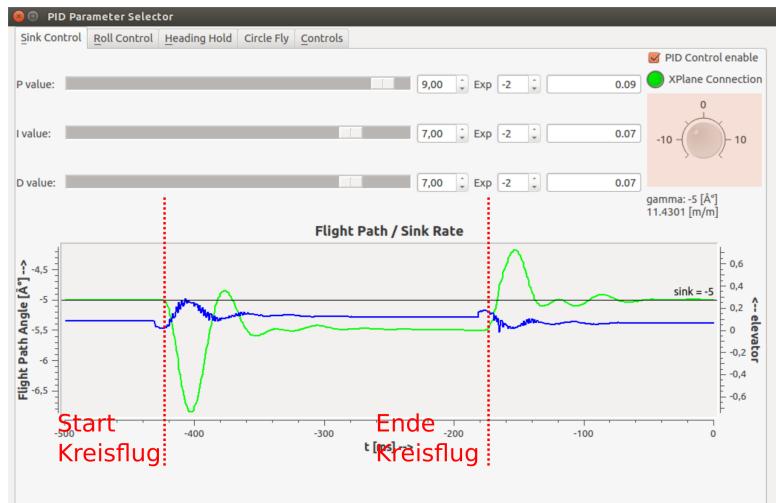


Abbildung 6.12: Abweichungen des gewünschten Gleitwinkels beim Einflug in den Kreis- $\gamma_{c,soll} = -5.5^\circ$ und bei Rückkehr in den Geradeausflug $\gamma_{s,soll} = -5.0^\circ$

die Sink-Control eine heftige Störung, die in der Summe den Gleitwinkel etwas steiler werden lässt. Demgegenüber gibt es beim Übergang aus dem Kreis- in den Geradeausflug einen deutlichen Überschwinger nach oben, der den Gleitwinkel im nachfolgenden Straight-Segment leicht anhebt. Dieser Überschwinger ist jedoch kleiner als der Unterschwinger beim Einflug in den Kreis und kann daher in der Summe die Abweichung nicht vollständig kompensieren. Weitere Untersuchungen müssen zeigen, ob es praktikabel ist, einen konstanten, von der Gesamtfluglänge unabhängigen Höhenfehler zur Korrektur anzunehmen, da die Segmentübergänge in jedem Anflug gleich oft vorkommen: Es gibt zwei Unterschwinger zum Kreisstart und zwei Überschwinger zum Kreisende.

GESCHWINDIGKEITEN UND FLUGDAUERN Bei der Ermittlung der zu verwendenden Gleitwinkel wurden auch die Geschwindigkeiten in den verschiedenen Flugsituations aufgezeichnet. Zu Beginn der Aufzeichnung befindet sich das Flugzeug zwar in einem stabilen, jedoch nicht stationären Flugzustand, da relativ kurz vorher ein Zustandsübergang stattgefunden hat, der zu einer Störung in der Geschwindigkeit führt.

Bei den Flugtests wurde in der Regel das Flugzeug „per Befehl“ auf eine bestimmte Ausgangshöhe angehoben in der andere Luftwiderstands- und Auftriebsverhältnisse herrschen als in der Fluglage davor. In einer realen Notsituation würde eine solche Störung wohl vom Übergang aus dem Motorflug in den Gleitflug dominiert werden, der darin besteht, dass der Schub bei Ausfall des Motors instantan wegfällt während die Flugzeuggeschwindigkeit nur langsam abnimmt. Eine weitere typische Störung ist

der Übergang vom Geradeaus- in den Kreisflug und zurück, bei dem sowohl der Roll- als auch der Gleitwinkel angepasst wird.

Wie sich schon bei der Ermittlung der Reglerparameter gezeigt hat, reagiert das Flugzeug um seine Pitch-Achse relativ träge (siehe Abbildung 6.1). Damit ist auch die Änderung der Geschwindigkeit im Vergleich zu anderen Parametern sehr langsam und je nach Anfangsbedingung dauert es entsprechend lang, bis die Geschwindigkeit nach einer Störung den von der Gleitpolare vorausgesagten stationären Wert annimmt.

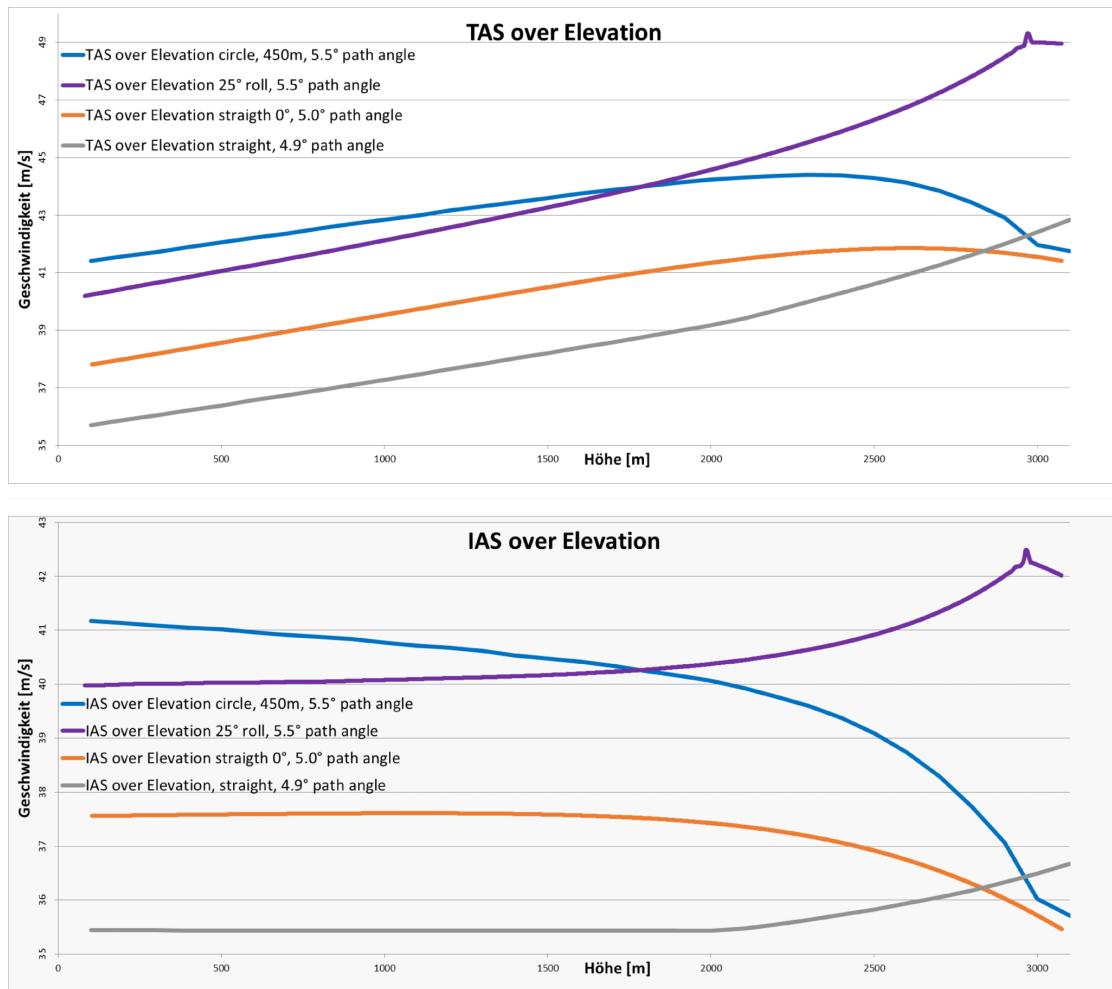


Abbildung 6.13: Geschwindigkeitsaufzeichnungen bei konstanten Gleitwinkeln über der Höhe

In Abbildung 6.13 sind Aufzeichnungen der TAS und der IAS zu sehen, die bei der Ermittlung der zu verwendenden Gleitwinkel aufgezeichnet wurden. Trotz der Laborsituation der Flugtests, in denen das Flugzeug vor Aufzeichnung in der Regel ausreichend Zeit hatte eine stabile Fluglage einzunehmen, fällt auf, dass es sehr lang

dauert, bis das Flugzeug tatsächlich seine stationäre Geschwindigkeit erreicht, die zur Gleitpolare passt.

Die Abbildungen enthalten neben der als Skala verwendeten Höhe implizit auch eine Zeitskala: Der Flug hat bei etwas über 3000m rechts angefangen und wurde knapp über dem Boden auf der linken Seite beendet. Die Flugdauer dafür beträgt je nach Gleitwinkel zwischen elf und 14 Minuten. Die genauen Flugdauern, die pro 100m Höhenverlust benötigt werden sind in Tabelle 6.4 eingetragen.

Tabelle 6.4: Sinkdauern für 100m Höhenverlust mit verschiedenen Parametern

Elevation [m]	time [s] straight $\gamma = -4.9^\circ$	time [s] straight $\gamma = -5.0^\circ$	time [s] $r = 450\text{ m}$ $\varphi = 25^\circ$ $\gamma = -5.5^\circ$	time [s] $\gamma = -5.5^\circ$
3000	27.6	27.6	24.2	20.9
2900	28.1	27.3	23.5	21.4
2800	28.1	27.6	23.6	21.9
2700	28.4	27.4	22.4	22.4
2600	28.6	27.3	23.6	22.4
2500	28.8	27.3	23.3	22.6
2400	29.1	27.5	23.5	22.9
2300	29.3	27.3	22.5	23.3
2200	29.6	27.6	22.4	23.2
2100	29.6	27.8	22.4	23.2
2000	29.8	27.8	23.6	23.4
1900	30.2	27.8	23.4	23.4
1800	30.2	27.9	23.7	23.7
1700	30.4	28.0	23.4	23.7
1600	30.4	26.9	23.5	24.1
1500	29.7	28.3	23.8	24.3
1400	30.5	28.5	24.2	23.6
1300	30.9	28.6	23.8	24.1
1200	31.1	28.7	24.2	23.9
1100	31.3	28.8	23.8	24.4
1000	30.5	29.0	23.3	24.7
900	31.6	28.0	23.3	24.9
800	31.7	29.2	23.6	24.4
700	31.8	29.5	24.0	25.2
600	31.1	28.5	24.7	24.2
500	32.2	29.7	24.7	24.7
400	32.3	29.9	24.9	24.3
300	32.5	30.0	24.9	25.6
overall time [s]:		845	790	662
				661

In den Abbildungen ist zu erkennen, dass die Annahme einer konstanten IAS bei konstantem Gleitwinkel γ und konstantem Rollwinkel φ nach Erreichen eines stationären Zustandes annähernd gültig ist: Zum Ende des Fluges, nach einer entsprechend langen Stabilisierungsphase verläuft die Kurve der IAS horizontal. Ein wenig komplizierter ist die Situation bei der TAS, bei der den großen Übergangszeitkonstanten zum Erreichen einer stationären Geschwindigkeit noch die Effekte durch die Lufdruckveränderungen aufgrund der Höhe überlagert sind.

Bei der Kurve für den Kreisflug mit konstantem Radius, und damit abnehmendem φ , kommt dazu auch noch der geringer werdende Rollwinkel in niedrigeren Höhen.

Man sieht in Abbildung 6.13, dass entsprechend die IAS in niedrigen Höhen zunimmt. Für die resultierende TAS ergibt sich damit eine Überlagerung zweier gegenläufiger Effekte: Mit abnehmender Höhe wird die TAS in Bezug auf die IAS geringer, während gleichzeitig die zugrundeliegende IAS mit abnehmendem Rollwinkel zunimmt.

Leider konnten noch nicht ausreichend Experimente durchgeführt werden, um ein Modell zur Vorhersage der zu erwartenden Geschwindigkeiten abzuleiten. Es ist jedoch schon aus diesen grundlegenden Betrachtungen heraus klar geworden, dass die in [Klein u. a., 2018, equ. 9] getroffene Annahme zweier konstanter Geschwindigkeiten v_c und v_s zur Voraussage der Geschwindigkeit im Kreis- bzw. Geradeausflug nicht zu halten ist.

Offensichtlich ist für eine möglichst genaue Voraussage der zu erwartenden Flugdauer ein komplexeres Modell zu entwickeln.

7 CONCLUSIO: ERGEBNISSE UND AUSBLICK

Mit der vorliegenden Arbeit wurde ein Autopilot entwickelt, mit dem sich Notlandeanflüge im Air-Frame simulieren lassen. Die abzufliegenden Bahnen basieren auf Dubins-Kurven, die aus einer Abfolge von Kreissegmenten und geraden Streckenabschnitten bestehen. Über eine Variation der Länge der geraden Segmente lässt sich bei ausgeregeltem Gleitwinkel der Höhenverlust während des Anflugs passgenau einstellen.

Zur Fluglageregelung wurden Regler entworfen und parametriert, mit denen sich über das Höhen- und das Querruder des Flugzeugs der Gleitwinkel und der Kurvenflug stabil ausregeln lassen. Weitere übergeordnete Regler sind in der Lage die vorkommenden Bahnsegmente (LRS) der Dubins-Pfade zu tracken und als Sequenz abzufliegen.

Der erstellte Autopilot arbeitet im sogenannten Air-Frame, in dem die geplanten Bahnen relativ zur umgebenden Luft definiert sind und sich daher mit dem Wind verschieben. Dieses Bezugssystem hat den großen Vorteil, dass die verwendeten Primitive der Dubins-Pfade nicht im Vorfeld transformiert werden müssen. Es kann weiter mit Kreissegmenten und Tangenten gerechnet werden, anstatt diese in Trochoide und Verbindungsstücke mit verändertem Winkel umzurechnen.

Bei der Überprüfung des Autopiloten konnte experimentell verifiziert werden, dass die Genauigkeit der Zielerreichung, unabhängig von Windeffekten, sehr gut ist. Bei einer Reihe von Flugtests im Simulator wurde das gewünschte Notlandefeld in allen Fällen bis auf eine Abweichung von wenigen Metern erreicht. Als systematischer Fehler wurde eine geringe Höhenabweichung nach unten festgestellt, die wahrscheinlich auf Übergangseffekte zwischen den einzelnen Dubins-Pfadsegmenten zurückzuführen ist.

Während der Validierung wurde neben der interaktiven Bedienmöglichkeit des Autopiloten eine Möglichkeit zur Control Automation entwickelt, so dass sich langwierige Flugtests weitgehend unbeaufsichtigt durchführen lassen. Der aktuelle Ausbau-stand richtet sich nach den Anforderungen der bisher durchgeführten Experimente, ist aber auf Erweiterungen ausgelegt.

Die Autopilotsoftware wurde von Beginn modular ausgelegt, damit Erweiterungen in Zukunft leicht möglich sind. Auch soll es leicht möglich sein Teile der Software –insbesondere die Regler zur Fluglageregelung und zur Bahnverfolgung– zu entnehmen und zum Beispiel in einer künftigen embedded Implementierung weiter zu verwenden.

Für den zukünftigen Ausbau des Systems zu einem vollautomatischen Emergency Landing Assistant (ELA) sind weitere Schritte notwendig:

Die wichtigste und zugleich komplexeste Fragestellung dürfte die Entwicklung eines Modells zur Voraussage des Windintegrals sein. Falls zu Beginn der Berechnung das gesamte Integral der Windgeschwindigkeit über den Anflug bekannt ist, können die Effekte der Windverblasung leicht neutralisiert werden: Dazu wird die gewünschte Zielposition im Vorfeld um genau dieses Integral entgegengesetzt zur vorausgesagten Translation durch den Wind verschoben. Diese Verschiebung entspricht genau einer Transformation der im Earth-Frame festen Zielposition in den mit dem Wind bewegten Air-Frame.

Unter der vereinfachenden Annahme konstanten Windes lässt sich das Windintegral aus einer genauen Voraussage der Anflugdauer direkt berechnen. Einfluss auf die Anflugdauer haben offenbar die in dieser Untersuchung identifizierten Parameter:

- die Anfangsgeschwindigkeit
- das Höhenprofil des geplanten Anflugs
- die Veränderung der TAS gegenüber der IAS mit abnehmender Höhe
- die Höhe der Kreissegmente zur Berücksichtigung des sich ändernden Rollwinkels und der sich daraus ergebenden IAS-Änderung
- die Übergangseffekte beim Wechsel von Kreis- auf Geradeausflug und umgekehrt

Offensichtlich ist für eine möglichst genaue Voraussage der zu erwartenden Flugdauer ein komplexes Modell zu entwickeln. Aktuell ist noch unklar, ob die analytische Formulierung eines solchen Modells möglich ist, oder ob z. B. durch Training eines neuronalen Netztes mit (automatisierten) Flugtests bessere Ergebnisse erzielt werden können.

Gegenüber dieser komplexen Fragestellung scheinen weitere Verbesserungsmöglichkeiten des Autopiloten und der Bahnplanung einfacher zu implementieren:

Mit dem aktuell implementierten Bahnplanungsalgorithmus nach [Klein u. a., 2018] wird die Verlängerung der Bahn bei überschüssiger Höhe ausschließlich durch Verlängerung der S-Segmente des Dubins-Pfads erreicht. Bei entsprechend großem Höhenüberschuss entfernt sich das verunfallte Flugzeug zunächst sehr weit vom gewünschten Notlandefeld bevor es in einem sehr ausgedehnten Final Approach wieder in Richtung des ELF fliegt. In dieser Situation scheint es für die Bahnplanung wünschenswert auch die Möglichkeit des Kreisens zu implementieren, um überschüssige Höhe abzubauen ohne sich zu weit vom ELF zu entfernen. Neben dem Kreisen in der Nähe des ELF werden noch weitere Optimierungen vor allem hinsichtlich der Variation des zu benutzenden Kreisradius r in [Paul, Hole, Ztyek und Varela, 2017] vorgeschlagen. Die dort beschriebenen Anregungen sehen für ein praktisch anwendbares System sehr vielversprechend aus, obwohl sie die Komplexität weiter erhöhen.

Aktuell sind die Gleitwinkel in allen Geraden und in allen Kreissegmenten als konstant und nahe am best glide angenommen. Dies ist auch notwendig, wenn das angesteuerte Notlandefeld sehr weit entfernt ist und kein großer Höhenüberschuss vorhanden ist. Bei näher gelegenen ELF sollten dagegen die Vorschläge aus [Izuta und Takahashi, 2017] berücksichtigt werden: Für den Final Approach ist ein steilerer Gleitwinkel vorzusehen, so dass bei unvorhergesehenen Störungen noch mehr Korrekturmöglichkeiten z. B. durch den Einsatz von Lande- oder Bremsklappen verbleiben. Das würde auch das Problem des minimal zu tiefen Anflugs lösen, das in der aktuellen Implementierung systematisch enthalten ist.

Darüber hinaus ist es sinnvoll den Gleitwinkel kurz vor dem Aufsetzen flacher auszuregeln, da die aktuell gewählten Winkel für den best glide im Geradeausflug zu recht harten Landungen führen. Dazu müsste eine weiteren Flugphase „Landing“ in die Software integriert werden.

Als letzte Verbesserung soll hier die Umstellung der einmaligen Bahnplanung zu Beginn des Anflugs auf eine kontinuierliche Berechnung zum Ausgleich unvorhergesehener Störungen genannt werden. Da sich der gesamte Anflug als sehr stabil erwiesen hat, ist die Neuberechnung im Abstand von einigen Sekunden ausreichend. Dafür muss der Bahnplanungsalgorithmus jedoch um die gegenläufigen Dubins-Pfade RSL und LSR erweitert werden. Über Pfade diesen Typs kann ein Flugzeug wieder „sanft“ auf den ursprünglichen Weg gebracht werden, auch wenn aufgrund der Fallunterscheidung in Gleichung (5.1) eine Unstetigkeitsstelle existiert. Die Bahnplanung würde damit von einer Steuerung in eine Regelung überführt.

Die entwickelte Software ist für die angesprochenen weiteren Untersuchungen und Verbesserungen vorbereitet. Pull-Requests mit Ergänzungen des Systems können über <https://github.com/opt12/DubinsPilot> zur Integration eingereicht werden.

Abschließend steht zu hoffen, dass diese Arbeit und die entwickelte Software einen Beitrag leisten in Zukunft einen vollautomatischen Emergency Landing Assistant (ELA) zu entwickeln, der die Sicherheit im Luftverkehr weiter verbessert.

LITERATUR

- [Allerton 2009] ALLERTON, David: *Principles of Flight Simulation*. 1. Aufl. [s.l.] : Wiley-Blackwell, 2009 (Aerospace Series). – ISBN 978-1-60086-703-3
- [Berger 2001] BERGER, Manfred: *Grundkurs der Regelungstechnik: mit Anwendung der Student Edition of MATLAB und SIMULINK ; mit 7 Tabellen*. Norderstedt : Books on Demand, 2001. – OCLC: 176655041. – ISBN 978-3-8311-0847-3
- [Bislins 2016] BISLINS, Walter: *Fluggeschwindigkeiten, IAS, TAS, EAS, CAS, Mach.* April 2016. – URL <http://walter.bislins.ch/blog/index.asp?page=Fluggeschwindigkeiten%2C+IAS%2C+TAS%2C+EAS%2C+CAS%2C+Mach>. – Zugriffsdatum: 2018-08-16
- [Blanchette 2008] BLANCHETTE, Jasmin: *C GUI Programming with Qt 4*. Second edition. Upper Saddle River, NJ Boston Indianapolis San Francisco New York Toronto : Prentice Hall, 2008 (Prentice Hall open source software development series). – ISBN 978-0-13-235416-5
- [Cessna Aircraft Company 2004] CESSNA AIRCRAFT COMPANY: *172S Skyhawk - Information Manual, Revision 5*. Juli 2004
- [Coombes u. a. 2014] COOMBES, Matthew ; CHEN, Wen-Hua ; RENDER, Peter: Reachability Analysis of Landing Sites for Forced Landing of a UAS. In: *Journal of Intelligent & Robotic Systems* 73 (2014), Januar, Nr. 1-4, S. 635–653. – ISSN 0921-0296, 1573-0409
- [Dubins 1957] DUBINS, L. E.: On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. In: *American Journal of Mathematics* 79 (1957), Juli, Nr. 3, S. 497. – ISSN 00029327
- [Eckstein u. a. 2018] ECKSTEIN, Felix ; SCHIFFMANN, Wolfram ; WITTICH, Björn: Emergency Landing Field Recognition Based on Elevation Data Using Parallel Processing. In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, September 2018, S. tbd.
- [FAA 2013] FAA: Chapter 3: Aerodynamics of Flight. In: *Glider Flying Handbook* Bd. FAA-H-8083-13A. URL https://www.faa.gov/regulations_policies/handbooks_manuals/aircraft/glider_handbook/. – Zugriffsdatum: 2018-09-05, 2013
- [FAA 2016] FAA: *Airplane Flying Handbook*. Bd. FAA-H-8083-3B. URL https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/airplane_handbook/. – Zugriffsdatum: 2018-08-18, November 2016

- [Izuta und Takahashi 2017] IZUTA, Shusuke ; TAKAHASHI, Masaki: Path Planning to Improve Reachability in a Forced Landing. In: *Journal of Intelligent & Robotic Systems* 86 (2017), Mai, Nr. 2, S. 291–307. – ISSN 0921-0296, 1573-0409
- [Klein u. a. 2018] KLEIN, Marius ; KLOS, Andreas ; LENHARDT, Jörg ; SCHIFFMANN, Wolfram: Moving Target Approach for Wind-Aware Flight Path Generation. In: *International Journal of Networking and Computing* 8 (2018), Nr. 2, S. 351–366. – ISSN 2185-2839, 2185-2847
- [Paul u. a. 2017] PAUL, Saswata ; HOLE, Frederick ; ZYTEK, Alexandra ; VARELA, Carlos A.: Flight Trajectory Planning for Fixed-Wing Aircraft in Loss of Thrust Emergencies. In: *arXiv:1711.00716 [cs]* (2017). – URL <http://arxiv.org/abs/1711.00716>. – Zugriffsdatum: 2018-09-30
- [Shkel und Lumelsky 2001] SHKEL, Andrei M. ; LUMELSKY, Vladimir: Classification of the Dubins Set. In: *Robotics and Autonomous Systems* (2001), S. 24
- [Tulapurkara 2012] TULAPURKARA, E. G.: *Flight Dynamics I - Airplane Performance*. 2012. – URL <https://nptel.ac.in/courses/101106041/3>. – Zugriffsdatum: 2018-08-16
- [Wang u. a. 2013] WANG, Yansen ; HUYNH, Giap ; WILLIAMSON, Chatt: Integration of Google Maps/Earth with Microscale Meteorology Models and Data Visualization. In: *Computers & Geosciences* 61 (2013), Dezember, S. 23–31
- [Williams] WILLIAMS, Ed: *Aviation Formulary V1.46*. – URL <http://www.edwilliams.org/avform.htm>. – Zugriffsdatum: 2018-08-16
- [Yechout 2014] YECHOUT, Thomas R.: *Introduction to Aircraft Flight Mechanics : Performance, Static Stability, Dynamic Stability, Classical Feedback Control, and State-Space Foundations*. 2. ed. Reston, Va. : AIAA, American Inst. of Aeronautics and Astronautics, 2014 (AIAA education series). – ISBN 978-1-62410-254-7

A ANHANG

A.1 Bedienungsanleitung

In den folgenden Abbildungen ist eine kurze Übersicht über die Bedienmöglichkeiten der GUI dargestellt. Über die Tabs in der obersten Leiste lässt sich das benötigte Bedienelement nach vorne holen.

Einstellung der Reglerparameter per

- Schieberegler
- Direkteingabe

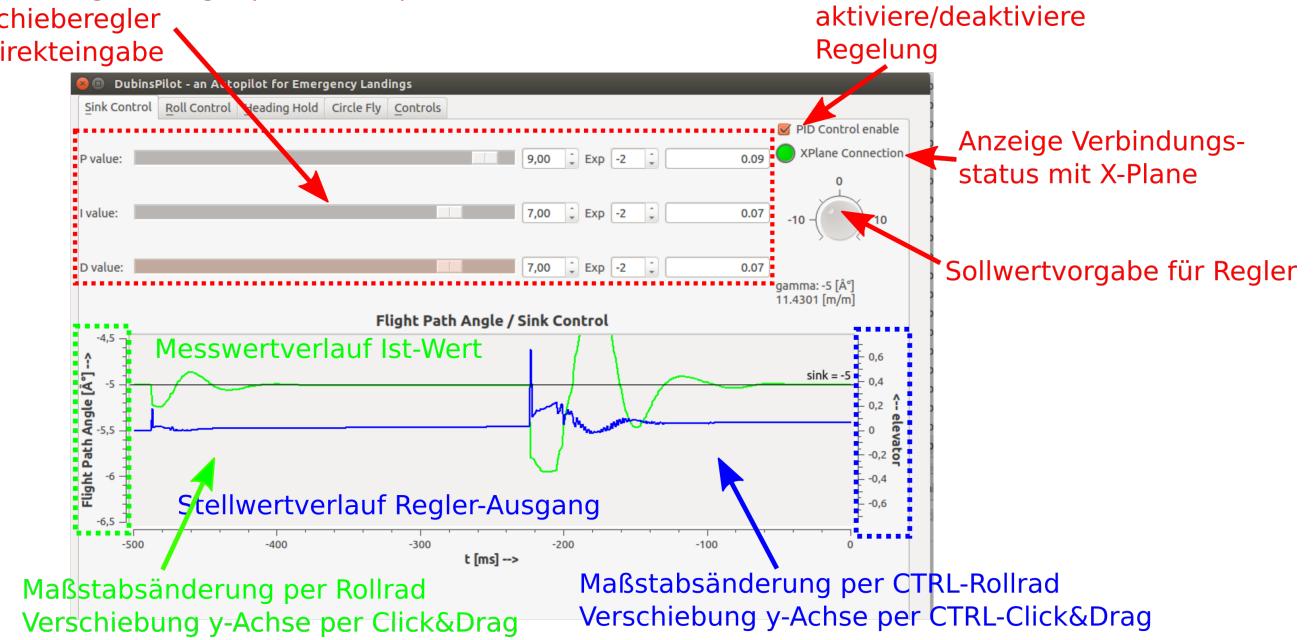


Abbildung A.1: Die Bedienmöglichkeiten eines Regler-Tabs.

Die Abbildung A.1 steht exemplarisch für alle vier über die GUI parametrierbaren Regler. Dargestellt ist lediglich der Regler für Sink-Control.

A-2 | A.2 Beispiel der erfassten statistischen Daten während der Bahnverfolgung

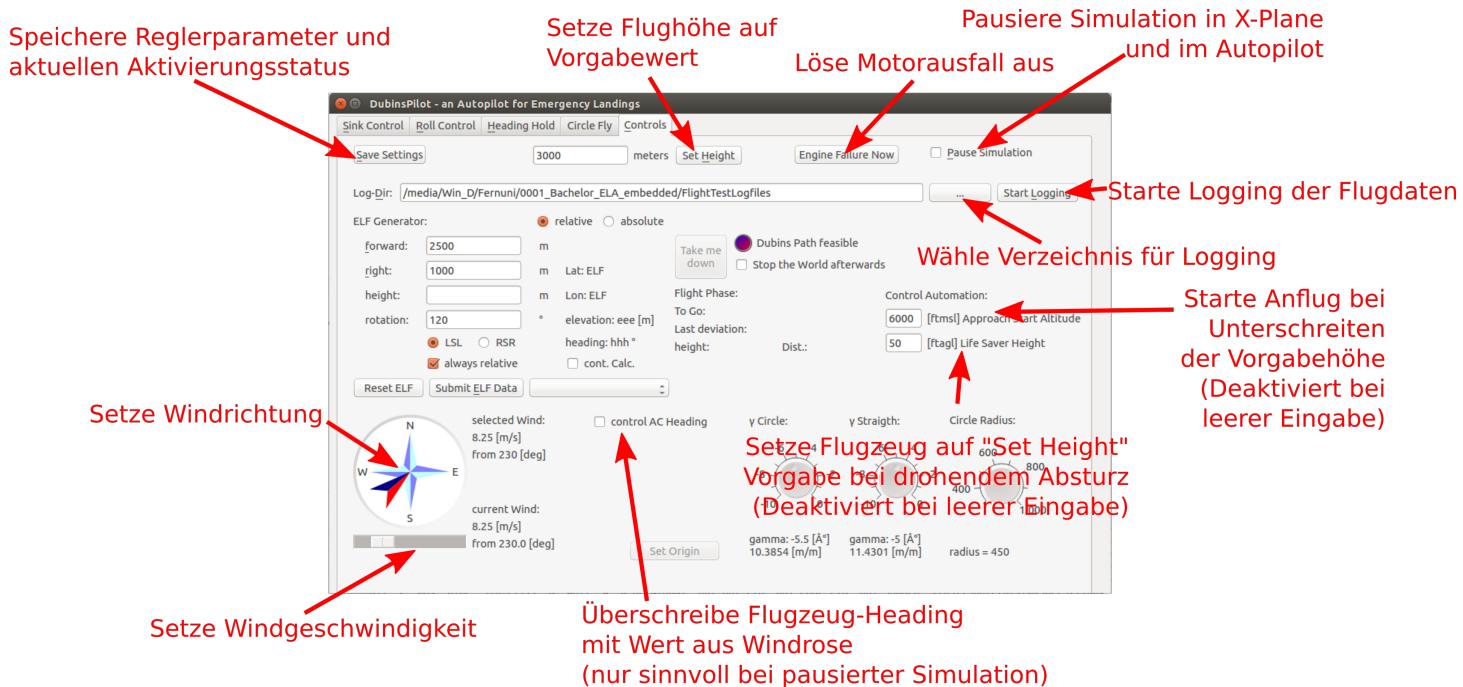


Abbildung A.2: Die Bedienmöglichkeiten im Control-Tab.

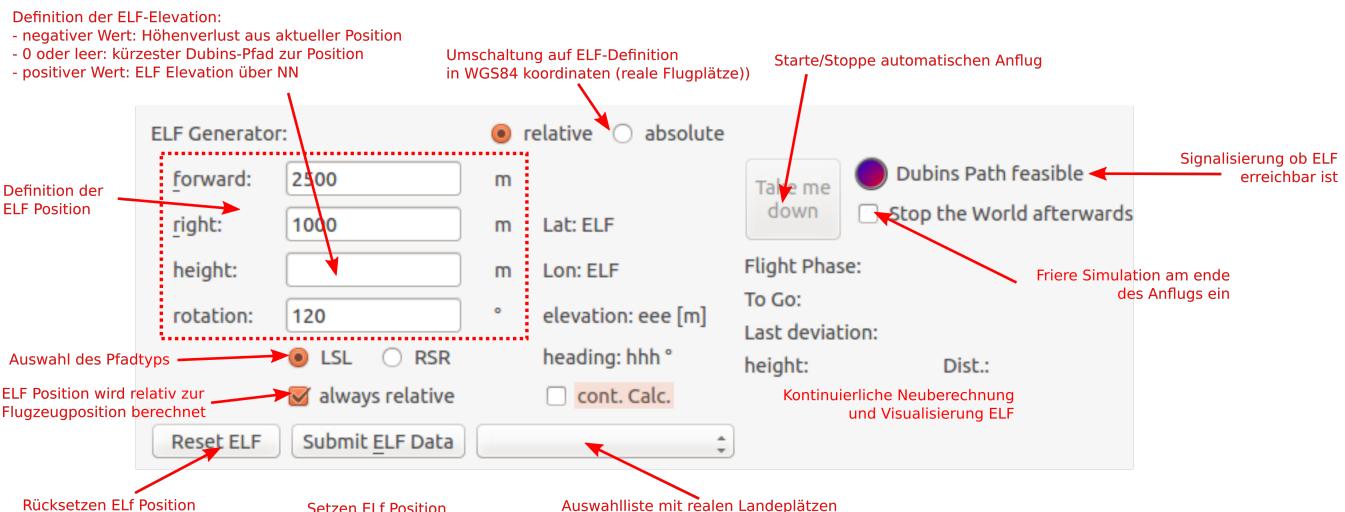


Abbildung A.3: Die Bedienmöglichkeiten zur ELF Definition und zur Anflugsteuerung Control-Tab.

A.2 Beispiel der erfassten statistischen Daten während der Bahnverfolgung

Während der Bahnverfolgung im Anflug werden eine Vielzahl von Messwerten aufgezeichnet und nach Ende des Anflugs in einem JSON-Objekt zusammengefasst. Hier

soll zunächst ein grober Überblick über die erfassten Parameter gegeben werden. Exemplarisch wird in Listing A.4 ein vollständiges Statistik-Objekt gezeigt.

Neben diesen Zusammenfassungen werden beim Logging auch vollständige CSV-Datensätze mit allen erfassten Mess- und Steuerwerten geschrieben. Diese lassen eine Detailauswertung über den Verlauf von Flügen zu.

Die erfassten Daten werden in vier verschiedene Äste gegliedert:

```
{
  "flownPath": { //...
},
  "pathSpecs": { //...
},
  "plannedPath": { //...
},
  "segStats": [ //...
]
}
```

Listing A.1: Überblick über die erfassten Daten-Kategorien während eines Anflugs.

Im Ast `pathSpecs` sind die Ausgangsdaten gespeichert, die bei der Berechnung des Anflugs verwendet wurden. Diese sind sowohl in Form absoluter Koordinaten im WGS84-System, als auch in Form relativer Koordinaten mit Bezug zur aktuellen Flugzeugposition erfasst:

```
{
  "pathSpecs": {
    "WGS84": {
      "end": {
        "altitude": 51.5,
        "heading": 271.0,
        "latitude": 52.45407415101304,
        "longitude": 9.709392786026001
      },
      "start": { //...
      }
    },
    "forward": 4255.7917472106965,
    "heightLoss": -690.6814575195313,
    "pathType": "LSL",
    "right": 4795.137721050784,
    "rotation": -88.7686767578125
  }
}
```

Listing A.2: Überblick über die erfassten Pfadspezifikationen zur Berechnung des Anflugs.

Im Ast `flownPath` sind die während des Anflugs tatsächlich erreichten Werte für verschiedene Kategorien aufgeschlüsselt. Dabei sind die Daten sowohl im Bezugssystem des Air-Frame als auch im Earth-Frame gegeben. Viele der zusammengefassten Parameter lassen sich auch je nach Pfadsegment aufgeschlüsselt weiter auswerten.

```
{
  "flownPath": {
    "airFrame": {
      "deviation": {
        "heading": 0.309722900390625,
        "x": -0.21070488261723835,
        "y": -1.6190991718747683,
        "z": 1.8309324068800528
      },
      "glideAngle": {
        "glideAngleCircle": 5.645999011669787,
        "glideAngleStraight": 4.964718956902967
      },
      "pathLength": 7823.511166460165,
      "pathLengthDetails": {
        "pathLengthCircleIn": 70.99194369681943,
        "pathLengthCircleOut": 566.553543319907,
        "pathLengthStraightFinal": 2561.2242698389537,
      }
    }
  }
}
```

A-4 | A.2 Beispiel der erfassten statistischen Daten während der Bahnverfolgung

```

        "pathLengthStraightTangential": 4624.741359790337
    },
    "angleDetails": {
        "angleCircleIn": -4.722900390625,
        "angleCircleOut": -72.41864013671875
    },
    "earthFrame": { //...
},
"flightTime": 212.03700000000003,
"flightTimeDetails": { //...
},
"heightLoss": 687.2606811523438,
"heightLossDetails": { //...
},
"totalAngle": -87.11480712890625
}
}

```

Listing A.3: Auszug aus den während des Anflugs gewonnenen Daten.

Im vollständigen Datensatz befinden sich noch eine Vielzahl weiterer Parameter. Je nach Fragestellung kann man die benötigten Parameter leicht aus diesem Objekt herausfiltern.

```

{
    "flownPath": {
        "airFrame": {
            "deviation": {
                "heading": 0.309722900390625,
                "x": -0.21070488261723835,
                "y": -1.6190991718747683,
                "z": 1.8309324068800528
            },
            "glideAngle": {
                "glideAngleCircle": 5.645999011669787,
                "glideAngleStraight": 4.964718956902967
            },
            "pathLength": 7823.5111166460165,
            "pathLengthDetails": {
                "pathLengthCircleIn": 70.99194369681943,
                "pathLengthCircleOut": 566.553543319907,
                "pathLengthStraightFinal": 2561.2242698389537,
                "pathLengthStraightTangential": 4624.741359790337
            }
        },
        "angleDetails": {
            "angleCircleIn": -4.722900390625,
            "angleCircleOut": -72.41864013671875
        },
        "earthFrame": {
            "deviation": {
                "heading": 0.309722900390625,
                "x": -0.21070488258102482,
                "y": -1.6190991716867078,
                "z": 1.8309324079101383
            },
            "glideAngle": {
                "glideAngleCircle": 5.645999011669787,
                "glideAngleStraight": 4.964718956902967
            },
            "pathLength": 7823.5111166460165,
            "pathLengthDetails": {
                "pathLengthCircleIn": 70.99194369681943,
                "pathLengthCircleOut": 566.553543319907,
                "pathLengthStraightFinal": 2561.2242698389537,
                "pathLengthStraightTangential": 4624.741359790337
            }
        },
        "windDisplacement": {
            "Heading": 90.0,
            "Length": 0.0
        }
    },
    "flightTime": 212.03700000000003,
    "flightTimeDetails": {
        "timeCircleIn": 1.941,
        "timeCircleOut": 15.0,
    }
}

```

A.2 Beispiel der erfassten statistischen Daten während der Bahnverfolgung | A-5

```

        "timeStraightFinal": 70.163,
        "timeStraightTangential": 124.933
    },
    "heightLoss": 687.2606811523438,
    "heightLossDetails": {
        "heightLossCircleIn": 7.6422119140625,
        "heightLossCircleOut": 55.386474609375,
        "heightLossStraightFinal": 219.15020751953125,
        "heightLossStraightTangential": 405.081787109375
    },
    "totalAngle": -87.11480712890625
},
"pathSpecs": {
    "WGS84": {
        "end": {
            "altitude": 51.5,
            "heading": 271.0,
            "latitude": 52.45407415101304,
            "longitude": 9.709392786026001
        },
        "start": {
            "altitude": 742.1814575195313,
            "heading": 359.7686767578125,
            "latitude": 52.409515380859375,
            "longitude": 9.769134521484375
        }
    },
    "forward": 4255.7917472106965,
    "heightLoss": -690.6814575195313,
    "pathType": "LSL",
    "right": 4795.137721050784,
    "rotation": -88.7686767578125
},
"plannedPath": {
    "angleDetails": {
        "angleCircleIn": -13.498297136846304,
        "angleCircleOut": -75.27037962096608
    },
    "angleTotal": -88.76867675781239,
    "glideAngleAirFrame": {
        "glideAngleCircle": -5.500000000000001,
        "glideAngleStraight": -5.0
    },
    "heightLoss": 690.6899381984018,
    "heightLossDetails": {
        "heightLossCircleIn": 10.291228330487115,
        "heightLossCircleOut": 56.94705503718677,
        "heightLossFinal": 221.35900977135884,
        "heightLossTangential": 404.16140705047434
    },
    "pathLengthAirFrame": 7824.492819076261,
    "pathLengthAirFrameDetails": {
        "pathLengthCircleIn": 106.01537780272122,
        "pathLengthCircleOut": 591.1721791253549,
        "pathLengthStraightFinal": 2530.2419939740475,
        "pathLengthStraightTangential": 4597.063268174137
    }
},
"segStats": [
{
    "airFrame": {
        "deviation": {
            "heading": -7.989386004033804,
            "x": 9.509694090927272,
            "y": -26.446031245108003,
            "z": -0.21100301416060674
        },
        "glideAngle": 6.1441728814577985,
        "heightLoss": 7.6422119140625,
        "pathLength": 70.99194369681943
    },
    "angleFlown": -4.722900390625,
    "earthFrame": {
        "deviation": {
            "heading": -7.989386004033804,
            "x": 9.509694090927272,
            "y": -26.446031245108003,
            "z": -0.21100301416060674
        }
    }
}
]

```

A-6 | A.2 Beispiel der erfassten statistischen Daten während der Bahnverfolgung

```
        "glideAngle": 6.1441728814577985,
        "heightLoss": 7.6422119140625,
        "pathLength": 70.99194369681943
    },
    "flightTime": 1.941,
    "segmentName": "Circle In",
    "windDisplacement": {
        "x": 0.0,
        "y": 0.0,
        "z": 0.0
    }
},
{
    "airFrame": {
        "deviation": {
            "heading": -0.16574708801817906,
            "x": -0.5963652872864966,
            "y": 1.7376460662056017,
            "z": -1.47543763322036
        },
        "glideAngle": 5.005770497892519,
        "heightLoss": 405.081787109375,
        "pathLength": 4624.741359790337
    },
    "angleFlown": -6.984527587890625,
    "earthFrame": {
        "deviation": {
            "heading": -0.16574708801817906,
            "x": -0.5963652874012293,
            "y": 1.737646066221237,
            "z": -1.4754376332323786
        },
        "glideAngle": 5.005770497892519,
        "heightLoss": 405.081787109375,
        "pathLength": 4624.741359790337
    },
    "flightTime": 124.933,
    "segmentName": "Straight Tangential",
    "windDisplacement": {
        "x": 0.0,
        "y": 0.0,
        "z": 0.0
    }
},
{
    "airFrame": {
        "deviation": {
            "heading": -3.011871337890625,
            "x": 31.478572615524037,
            "y": 6.261006183047414,
            "z": -0.12264016201330463
        },
        "glideAngle": 5.583513385717354,
        "heightLoss": 55.386474609375,
        "pathLength": 566.553543319907
    },
    "angleFlown": -72.41864013671875,
    "earthFrame": {
        "deviation": {
            "heading": -3.011871337890625,
            "x": 31.478572615524037,
            "y": 6.261006183047414,
            "z": -0.12264016201330463
        },
        "glideAngle": 5.583513385717354,
        "heightLoss": 55.386474609375,
        "pathLength": 566.553543319907
    },
    "flightTime": 15.0,
    "segmentName": "Circle Out",
    "windDisplacement": {
        "x": 0.0,
        "y": 0.0,
        "z": 0.0
    }
},
{
    "airFrame": {
        "deviation": {
```

```

        "heading": 0.309722900390625,
        "x": -0.21070488261723835,
        "y": -1.6190991718747683,
        "z": 1.8309324068800528
    },
    "glideAngle": 4.8905802541767605,
    "heightLoss": 219.15020751953125,
    "pathLength": 2561.2242698389537
},
"angleFlown": -2.988739013671875,
"earthFrame": {
    "deviation": {
        "heading": 0.309722900390625,
        "x": -0.21070488258102482,
        "y": -1.6190991716867078,
        "z": 1.8309324079101383
    },
    "glideAngle": 4.8905802541767605,
    "heightLoss": 219.15020751953125,
    "pathLength": 2561.2242698389537
},
"flightTime": 70.163,
"segmentName": "Final Approach",
"windDisplacement": {
    "x": 0.0,
    "y": 0.0,
    "z": 0.0
}
}
]
}
}

```

Listing A.4: Vollständiger erfasster Statistik-Datensatz für einen Anflug auf eine reale Landebahn.

A.3 Datenaustausch mit X-Plane

Zur UDP-Kommunikation und zur Umwandlung generischer Datentypen in das X-Plane-DataRef-Format²⁰ kommt als Basis die Bibliothek `libXPlane-UDP-Client` des GitHub-Users **Shahada Abubakar** zum Einsatz. An dieser Bibliothek mussten kleiner Änderungen und Fixes vorgenommen werden, damit auch das Debugging unterstützt wird. Daher stellt der Autor einen Fork dieser Bibliothek unter <https://github.com/opt12/libXPlane-UDP-Client> bereit.

In diesem Abschnitt werden die DataRefs angegeben, über die Autopilot Daten mit dem Simulator austauscht. Die für die Flugregelung und den Autopiloten unbedingt erforderlichen Daten sind entsprechend gekennzeichnet. Zusätzliche Daten werden lediglich dazu verwendet weitere Informationen in den Log-Dateien zur weiteren Auswertung zur Verfügung zu haben.

²⁰ Das Datenformat der DataRefs ist in der Datei `/Instructions/X-Plane SPECS from Austin/Exchanging Data with X-Plane.rtf` im Installationsverzeichnis von X-Plane beschrieben.

Tabelle A.1: Bei X-Plane abonnierte DataRefs

usage:	Name	Type	Writable	Units	Description
Logging	sim/flightmodel/position/indicated_airspeed	float	y	kias	Air speed indicated - this takes into account air density and wind direction
Autopilot	sim/flightmodel/position/true_airspeed	float	n	meters/sec	Air speed true - this does not take into account air density at altitude!
Logging	sim/flightmodel/position/groundspeed	float	n	meters/sec	The ground speed of the aircraft
Logging	sim/flightmodel/position/vh_ind_fpm	float	y	fpm	VVI (vertical velocity in feet per second)
Autopilot	sim/flightmodel/position/vh_ind	float	n	meters/sec	VVI (vertical velocity in meters per second)
Logging	sim/flightmodel/position/vpath	float	n	degrees	The pitch the aircraft actually flies. (vpath+alpha=theta)
Logging	sim/flightmodel/position/alpha	float	n	degrees	The pitch relative to the flown path (angle of attack)
Logging	sim/flightmodel/position/true_theta	float	n	degrees	The pitch of the aircraft relative to the earth precisely below the aircraft
Autopilot	sim/flightmodel/position/true_phi	float	n	degrees	The roll of the aircraft relative to the earth precisely below the aircraft
Autopilot	sim/flightmodel/position/true_psi	float	n	degrees	The heading of the aircraft relative to the earth precisely below the aircraft - true degrees north, always
Logging	sim/flightmodel/position/elevation	double	n	meters	The elevation above MSL of the aircraft
Autopilot	sim/flightmodel/position/latitude	double	n	degrees	The latitude of the aircraft
Autopilot	sim/flightmodel/position/longitude	double	n	degrees	The longitude of the aircraft
Logging	sim/cockpit2/gauges/indicators/altitude_ft_pilot	float	n	feet	Indicated height, MSL, in feet, primary system, based on pilots barometric pressure input.
Logging	sim/flightmodel/position/y_agl	float	n	meters	AGL
Logging	gauges/indicators/roll_electric_deg_pilot	float	n	degrees	Indicated roll, in degrees, positive right. Source: electric gyro. Side: Pilot
Autopilot	sim/weather/wind_direction_degt	float	n	[0-359]	The effective direction of the wind at the plane's location.
Autopilot	sim/weather/wind_speed_kt	float	n	kts	>= 0 The effective speed of the wind at the plane's location.

Tabelle A.2: Von der Autopilot-Software überschriebene DataRefs

Autopilot usage:	Name	Type	Writable	Units	Description
Setup	sim/operation/override override_joystick_pitch	int	y	boolean	Override just pitch control. Use yoke_pitch_ratio.
Autopilot	sim/joystick/yoke_pitch_ratio	float	y	[-1..1]	The deflection of the joystick axis controlling pitch. Use override_joystick or override_joystick_pitch
Setup	sim/operation/override override_joystick_roll	int	y	boolean	Override just roll control. Use yoke_roll_ratio.
Autopilot	sim/joystick/yoke_roll_ratio	float	y	[-1..1]	The deflection of the joystick axis controlling roll. Use override_joystick or override_joystick_roll
Setup	sim/weather/wind_altitude_msl_m[0..2]	float	y	meters	≥ 0 The center altitude of this layer of wind in MSL meters
Setup	sim/weather/wind_speed_kt[0..2]	float	y	kts ≥ 0	The wind speed in knots.
Setup	sim/weather/wind_direction_degt[0..2]	float	y	[0 - 360]	The direction the wind is blowing from in degrees from true north clockwise.
Sim-Control	sim/time/sim_speed	int	y	ratio	This is the multiplier for real-time...1 = real-time, 2 = 2x, 0 = paused, etc.
Sim-Control	sim/flightmodel/position/local_y	double	y	meters	The location of the plane in OpenGL coordinates

In Tabelle A.1 sind alle abonnierten DataRefs gelistet. Die für die Flugregelung verwendeten sind mit der Kennzeichnung „Autopilot“ versehen. Die lediglich für die Evaluation verwendeten mit dem Wort „Logging“.

In Tabelle A.2 sind die DataRefs angegeben, die durch das Evaluationssystem manipuliert werden. Der Autopilot überschreibt lediglich die beiden Eingabewerte für den Joystick in den Achsen „Pitch“ und „Roll“. Für das Simulationssetup sind weitere Werte zu setzen, die mit „Setup“ gekennzeichnet sind. Die beiden Werte mit der Kennzeichnung „Sim-Control“ werden verwendet um während der SW-Erprobung den Simulator zu pausieren, bzw. das Flugzeug auf eine sichere Höhe anzuheben.

X-Plane hat bei der Angabe des Windes einen Bug, der zu berücksichtigen ist: Der Wert, der in der Dataref `sim/weather/wind_speed_kt` von X-Plane ausgegeben wird ist in $\frac{m}{s}$ angegeben und nicht in Knoten. Wenn man den Wind dagegen über die Datarefs `sim/weather/wind_speed_kt []` schreibt, so erwartet X-Plane einen Wert in Knoten.

A.4 SW-Architektur en Détail

Die gesamte Software besteht aus insgesamt lediglich sieben Objekten, die auf oberster Ebene in der `main`-Funktion einmal instantiiert werden. Diese halten gegenseitig keine Referenzen aufeinander, sind also nur lose gekoppelt. Der Informations- und Befehlaustausch der einzelnen Toplevel-Objekte erfolgt über den vom Framework Qt bereitgestellten „Signals & Slots“ Mechanismus. Dieser Mechanismus ermöglicht eine lose Kopplung der einzelnen Module und stellt ein definiertes Interface zwischen Ihnen bereit. Details dazu finden sich in [Blanchette, 2008].

Da der Signals&Slots Mechanismus nicht erlaubt synchrone Anfragen mit Rückgabewert an ein Objekt zu stellen, sondern lediglich Befehlsaufrufe in die EventQueue einsortiert, wurde für die zentrale Datenhaltung ein anderes Muster gewählt: Das Objekt `DataCenter` ist als sogenanntes Singleton-Objekt realisiert. Damit ist garantiert, dass es nur ein einziges Mal innerhalb des Programms existiert und jeder Interessent auf die gleichen konsistenten Daten zugreifen kann. Obwohl die Verwendung von Singleton-Objekten Gegenstand kontroverser Diskussionen ist, wurde hier entschieden es für die Datenhaltung einzusetzen, da die Vorteile bei Weitem überwiegen.

Die Kommunikation der Objekte untereinander ist in Abbildung A.4 dargestellt. Die tatsächlich ausgetauschten Signale sind in den Tabellen A.4 - A.18 angegeben. Falls in einer weitergehenden ELA-Implementierung in einer anderen Umgebung einzelne Module ersetzt werden sollen, so müssen lediglich die Signale dieser Schnittstelle bereitgestellt werden um ein einzelnes Modul nahtlos zu ersetzen.

Neben den gezeigten Toplevel-Objekten wurden für die Realisierung der Software natürlich noch eine Reihe weiterer Klassen implementiert. Diese sollen nur kurz in Tabelle A.3 genannt werden, ohne näher auf Implementierungs- und Verwendungsdetails einzugehen. Diese gehen aus den beigelegten Sourcen der Applikation hervor.

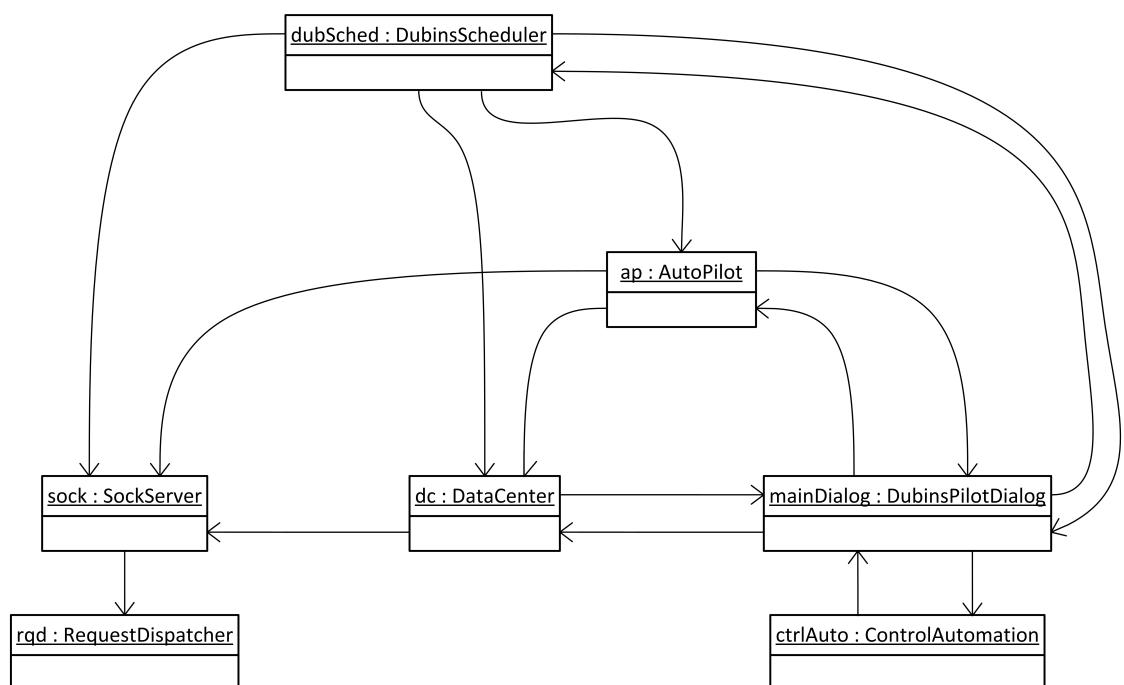


Abbildung A.4: Die in der main-Applikation erstellten Toplevel-Objekte und Signalwege.

Tabelle A.3: Wichtige Klassen der SW-Implementierung

AutoPilot	Die zentrale Klasse zur Flugregelung. Kontrolliert die implementierten Regler.
Controller	Innere Klasse in AutoPilot um Objekte vom Typ PIDControl inklusive Parametern aufzunehmen.
PIDControl	Implementierung eines generischen PID-Controllers.
ControlAutomation	Enthält Routinen um automatisiert Ereignisse auszulösen. Wird für die Automatisierung der Flight-Tests benutzt.
DataCenter	Das zentrale Singleton-Objekt zur Datenhaltung.
Dataset	Ein einzelner Datensatz, der vom DataCenter-Objekt verwaltet wird.
DubinsPath	Beschreibt einen Dubins-Pfad im Air-Frame. Die Berechnung der Pfade findet im Konstruktor statt.
DubinsScheduler	Die Ablaufsteuerung beim Path-Tracking eines Dubins-Pfades. Überwacht den Anflug und aktiviert die für das aktuelle Segment erforderlichen Flugphasen.
FlightPhase	Abstrakte Klasse zur Beschreibung einer Flugphase.
CirclePhase	Implementiert FlightPhase; Ablaufsteuerung um ein L- oder R-Segment des Dubins-Pfades abzufliegen. Überwacht den Flug und aktiviert dazu passend den Circle-Fly Regler.
StraightPhase	Implementiert FlightPhase; Ablaufsteuerung um ein S-Segment des Dubins-Pfades abzufliegen. Überwacht den Flug und aktiviert dazu passend den Heading-Hold Regler. In dieser Klasse ist auch der Localizer Interception Regler implementiert.
RunOutPhase	Implementiert FlightPhase; Ablaufsteuerung für die Psudo-Flugphase Run-Out. Beendet die Pfadverfolgung und erstellt die statistische Zusammenfassung.
SegmentStatistics	Verwaltet die Statistikdaten eines einzelnen Bahnsegments
Position	Hilfsklassen zum Verwalten von Positionsangaben. Stellt viele Hilfsfunktionen bereit um von WGS84-Koordinaten in kartesische Koordinaten umzurechnen.
Position_WGS84	Hilfsklassen zum Verwalten von Positionsangaben im WGS84-Format.
Position_Cartesian	Hilfsklassen zum Verwalten von Positionsangaben in kartesischen Koordinaten.
RequestDispatcher	Aktuell unbenutzt. Vorbereitung einer externen Fernsteuerung der Software von außen. Kann JSON-Objekte von SockServer entgegennehmen, auspacken und entsprechende Aktionen Triggeren.
SockServer	Öffnet einen POSIX-Socket als konkreten Kommunikationskanal zur Visualisierung. Leitet von extern empfangene Befehle an den RequestDispatcher weiter.

Tabelle A.4: connections from DubinsPilotDialog → AutoPilot

SIGNAL(sigPidParametersChanged(ctrlType, double, double))	neue Reglerparameter verfügbar
SIGNAL(sigCtrlActiveStateChanged(ctrlType, bool))	Regleraktivierung / Reglerdeaktivierung
SIGNAL(sigRequestedSetValueChanged(ctrlType, double))	neuer Sollwert für Regler
SIGNAL(sigCircleDirectionChanged(bool, double))	neue Drehrichtung für Circle-Fly

Tabelle A.5: connections from DubinsScheduler → AutoPilot

SIGNAL(sigCtrlActiveStateChanged(ctrlType, bool))	Regleraktivierung / Reglerdeaktivierung
SIGNAL(sigRequestedSetValueChanged(ctrlType, double, bool, bool))	neuer Sollwert für Regler
SIGNAL(sigCircleDirectionChanged(bool, double))	neue Drehrichtung für Circle-Fly

Tabelle A.6: connections from DubinsPilotDialog → DataCenter

SIGNAL(sigLoggingActiveStateChanged(bool, QFile *))	Logging aktiviert / deaktiviert
SIGNAL(sigSendXPDataRef(const char*, double))	sende geänderte Dataref an X-Plane
SIGNAL(sigSetElfLocation(double, double, double, double, pathTypeEnum))	setze geänderte ELF-Position (relativ zum Flugzeug)
SIGNAL(sigSetElfLocation(Position_WGS84, double, pathTypeEnum))	setze geänderte ELF-Position (absolute WGS84-Koordinaten)
SIGNAL(sigResetElf(void))	lösche aktuelles ELF
SIGNAL(sigSetSimulationPaused(bool))	pausiere Autopilot und Simulator
SIGNAL(sigFlightPathCharacteristicsChanged(double, double, double))	setze Parameter für den Dubins-Pfad (Gleitwinkel $\gamma_{c,s}$, Kreisradius r)

Tabelle A.7: connections from AutoPilot → DataCenter

SIGNAL(sigSendXPDataRef(const char*, double))	sende geänderte Dataref an X-Plane
SIGNAL(sigSendXPDataRef(const char*, bool))	sende geänderte Dataref an X-Plane

Tabelle A.8: connections from DataCenter → DubinsPilotDialog

SIGNAL(XPlaneConnectionChanged(bool))	zeige geänderteren Verbindungsstatus zum Simulator in der GUI an
SIGNAL(sigElfCoordsSet(Position_WGS84, double, bool))	zeige neue ELF-Position in der GUI an
SIGNAL(sigWindChanged(double, double))	zeige geänderten Wind in der GUI an

Tabelle A.9: connections from AutoPilot → DubinsPilotDialog

SIGNAL(sigAttachControllerCurve(ctrlType, QwtPlotCurve*))	zeige Plot der Reglerwerte in der GUI an
SIGNAL(sigReplotControllerCurve(ctrlType))	aktualisiere Plot der Reglerwerte in der GUI
SIGNAL(sigRequestTargetValue(ctrlType, double, bool))	zeige geänderten Sollwert für Regler in der GUI an
SIGNAL(sigSetControllerCheckButtons(ctrlType, bool, bool))	zeige geänderte Controlleraktivierung in der GUI an

Tabelle A.10: connections from SockServer → RequestDispatcher

SIGNAL(sigDispatchSockMessage(json))	aktuell unbenutzt; Vorbereitung für Fernsteuerung der Applikation von außen.
--------------------------------------	--

Tabelle A.11: connections from DataCenter → SockServer

SIGNAL(sigSocketSendData(std::string, int, json))	publiziere JSON Objekt nach außen
---	-----------------------------------

Tabelle A.12: connections from AutoPilot → SockServer

SIGNAL(sigSocketSendData(std::string, int, json))	publiziere JSON Objekt nach außen
---	-----------------------------------

Tabelle A.13: connections from DubinsPilotDialog → DubinsScheduler

SIGNAL(sigStartPathTracking(bool))	starte Pfadverfolgung
------------------------------------	-----------------------

Tabelle A.14: connections from DubinsScheduler → DubinsPilotDialog

SIGNAL(sigDisplayFlightPhase(QString, QString))	zeige aktuelle Flugphase in der GUI
SIGNAL(sigPathTrackingStatus(bool))	zeige aktuellen Status der Pfadverfolgung in der GUI
SIGNAL(sigPauseSimTriggered(bool))	klicke den "Pause"Button in der GUI

Tabelle A.15: connections from DubinsScheduler → DataCenter

SIGNAL(sigOutputPathTrackingStats(const json))	speichere neue Statistikdaten
--	-------------------------------

Tabelle A.16: connections from DubinsScheduler → SockServer

SIGNAL(sigSocketSendData(std::string, int, json))	publiziere JSON Objekt nach außen
---	-----------------------------------

Tabelle A.17: connections from DubinsPilotDialog → ControlAutomation

SIGNAL(sigLifeSaverHeightChanged(QString))	setze neuen Startwert für ControlAutomation
SIGNAL(sigApproachStartingAltitudeChanged(QString))	setze neuen Startwert für ControlAutomation

Tabelle A.18: connections from ControlAutomation → DubinsPilotDialog

SIGNAL(sigLifeSaverTriggered(void))	Führe Controlautomation Event über die GUI aus
SIGNAL(sigApproachStartingTriggered(void))	Führe Controlautomation Event über die GUI aus

A.5 Visualisierung

Zur Visualisierung wurde eine simple React-Web-App erstellt, die Positions- und Bahn-daten in eine Leaflet-Karte einblendet. Diese Daten werden von einem minimalen Express-Server auf einer REST-API bereitgestellt.

Um die Visualisierung zu nutzen, muss lediglich die Server-Komponente AutoViewer gestartet werden. Das geht mit einem einfachen `npm start` auf der Konsole im Verzeichnis `/AutoViewer`. Dieser Express-Server liefert auf `localhost:3001` die Web-App aus dem Verzeichnis `../AutoViewerClient/build` aus. Sie lässt sich einfach mit einem modernen Browser anzeigen.

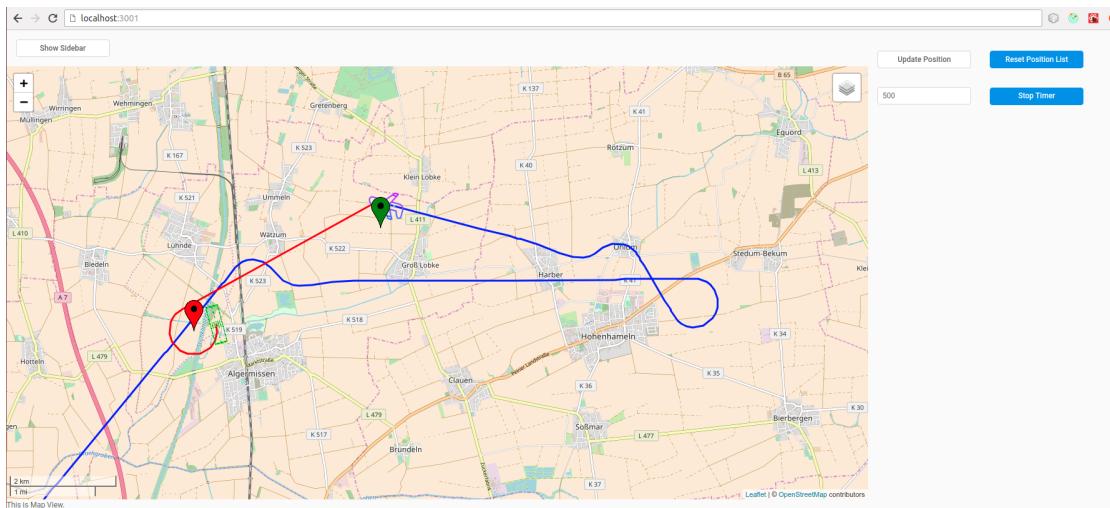


Abbildung A.5: Die sehr einfach gehaltene Web-App zur Darstellung der aktuellen und geplanten Flugbahnen in einer frei wählbaren Karte.

Die Express-App AutoViewer dient als Proxy, um die aktuellen Datensätze des Autopiloten aus einem POSIX-Socket entgegen zu nehmen und auf einer REST-API per HTTP zur Verfügung zu stellen. Sie hält dazu intern einen Datenspeicher als Cache mit den vom Autopiloten publizierten JSON-Objekten. Der AutoViewerClient fragt diese Daten Timer-gesteuert beim Server ab und stellt sie in der ausgewählten Karte dar. Standardmäßig ist die OpenStreetMaps-Karte eingestellt. Es können aber auch Karten und Satellitenbilder von Google ausgewählt werden. Für die meisten Abbildungen in dieser Karte wurde ein neutraler Hintergrund ohne Kartendarstellung gewählt, da die konkrete Position keine Rolle spielt.

Die in Abbildung A.5 dargestellte Web-App ist aktuell noch sehr rudimentär gehalten, kann aber leicht erweitert werden. Insbesondere kann darüber auch eine alternative Benutzeroberfläche für den Autopiloten realisiert werden. Dazu müsste die Kommunikation des AutoViewer mit dem Autopiloten zu einer Zwei-Wege-Kommunikation

erweitert werden. Die Autopilot-Software selber ist über Ihre Klasse RequestDispatcher auf die Entgegennahme externer Kommandos vorbereitet.

A.6 Installationsanleitung und externe Bibliotheken

Die Applikation wurde unter Ubuntu-14.04 mit GCC entwickelt, sollte aber durch die Verwendung des Qt-Frameworks auch unter Windows und OS X lauffähig sein. Leider konnte das aufgrund fehlender eigener Hardware noch nicht getestet werden.

Es werden einige externe Bibliotheken verwendet. Einige davon sind als sogenannte „Header-Only“-Libraries, bzw. als sehr kleine Quelltextpakete direkt in dem Unterverzeichnis /external enthalten. Dies sind:

- betterEnums: <https://github.com/aantron/better-enums.git>
- JSON for Modern C++: <https://github.com/nlohmann/json.git>
- XPlane UDP Client: <https://github.com/opt12/libXPlane-UDP-Client.git>
- QLedIndicator: <https://www.linux-apps.com/p/1132137/>

Desweiteren ist eine stark modifizierte Version der PID-Lib Arduino direkt in das /src Unterverzeichnis eingeflossen:

- PID-Lib Arduino: https://github.com/tcleg/PID_Controller

Als System-Bibliotheken wurden folgende Bibliotheken genutzt:

- PThreads; Diese Bibliothek ist Teil der GCC-Standard-Installation.²¹
- Funktionen aus sys/socket.h; Diese Funktionen sind Teil der GCC-Standard-Installation.²¹
- Qt 4.8: <https://doc.qt.io/archives/qt-4.8/>; Diese Bibliothek ist unter Ubuntu verfügbar über sudo apt-get install qt4-default.
- GeographicLib: <https://geographiclib.sourceforge.io/>; Diese Bibliothek ist unter Ubuntu verfügbar über sudo apt-get install libgeographic-dev.
- QWT-6.0.2: <https://sourceforge.net/projects/qwt/files/qwt/6.0.2/>; Diese Bibliothek muss manuell aus den Sourcen installiert werden.

²¹ Unter Windows muss dazu entweder MinGW oder Cygwin verwendet werden. Alternativ müssen diese POSIX-Konstrukte in native Windows Funktionen umgesetzt werden. Näheres dazu findet sich z. B. <https://stackoverflow.com/questions/2952733/using-sys-socket-h-functions-on-windows>.

Leider wurden in der Bibliothek QWT von der Version 6.0.2 auf die Version 6.1.x sogenannte *breaking changes* eingeführt, so dass aktuell noch die ältere Version 6.0.2 benutzt werden muss. Das erklärt auch die Verwendung von Qt 4 anstelle von Qt 5. Es ist geplant die Applikation auf GitHub demnächst auf die aktuellere Version von Qt 5 zusammen mit QWT-6.1 umzustellen.

Für die Visualisierungskomponente der Software wurde node.js verwendet. Um diese Applikation zu erstellen sind lediglich eine aktuelle Version von node.js sowie des Paketmanagers npm erforderlich. Die Installation der externen Bibliotheken wird dadurch automatisiert.

Die Quelltexte zur Übersetzung sind auf der beiliegenden CD enthalten. Hier ist auch unter /Software_Sources/DubinsPilot/README.md eine genaue Installationsanleitung zu finden.

Bequemer läuft die Installation jedoch direkt aus den GitHub-Repositories:

<https://github.com/opt12/DubinsPilot> und

<https://github.com/opt12/DubinsViewer>

Auf GitHub ist auch eine ausführliche Installationsanleitung hinterlegt.

ERKLÄRUNG

Name: Felix Eckstein
Matrikel-Nr.: 8161569
Fach: Informatik
Modul: Bachelorarbeit

Ich erkläre, dass ich die vorliegende Abschlussarbeit mit dem Thema

*Implementierung eines Notlandeassistenten auf Basis von Dubins-Kurven
unter Berücksichtigung des Windes*

selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich, inhaltlich oder sinngemäß entnommenen Stellen als solche den wissenschaftlichen Anforderungen entsprechend kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für Zeichnungen, Skizzen oder graphische Darstellungen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft und ausschließlich für Prüfungszwecke gespeichert wird. Außerdem räume ich dem Lehrgebiet das Recht ein, die Arbeit für eigene Lehr- und Forschungstätigkeiten auszuwerten und unter Angabe des Autors geeignet zu publizieren.

Hannover, den 30. September 2018

Dipl.-Ing. Dipl.-Wirt. Ing. Felix Eckstein