DLR-IB-FT-BS-2023-19

Detect & Avoid unter der Berücksichtigung von Hindernissen am Beispiel des tieffliegenden unbemannten Luftfahrzeugs "ALAADy-Demonstrator" (Masterarbeit)

Interner Bericht
Hochschulschrift

Autor: Matthias Guthörl

DLR Deutsches Zentrum für Luft- und Raumfahrt

Institutsbericht
IB 111-2023/19

Detect & Avoid unter der Berücksichtigung von Hindernissen am Beispiel des tieffliegenden unbemannten Luftfahrzeugs "ALAADy-Demonstrator" (Masterarbeit)

Matthias Guthörl

Institut für Flugsystemtechnik
Braunschweig

112	Seiten
48	Abbildungen
4	Tabellen
69	Referenzen

Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Flugsystemtechnik
Abteilung Unbemannte Luftfahrzeuge

Braunschweig, den 16.02.2023

[redacted]

Institutsdirektor:	Prof. Dr.-Ing. S. Levedag
Abteilungsleiter:	Johann C. Dauer
Betreuer:	Dr.-Ing Sebastian Benders
Verfasser:	Matthias Guthörl

Detect & Avoid unter der Berücksichtigung von Hindernissen am Beispiel des tieffliegenden unbemannten Luftfahrzeugs "ALAADy-Demonstrator"

Wissenschaftliche Arbeit zur Erlangung des Grades
M.Sc. Aerospace
an der TUM School of Engineering and Design der Technischen Universität München.

Betreut von Prof. Dr.-Ing. Markus Ryll
Professur für Autonomous Aerial Systems

Dr.-Ing. Sebastian Benders
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Institut für Flugsystemtechnik

Eingereicht von Matthias Guthörl
[redacted]

Eingereicht am Braunschweig, den 10.02.2023

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Braunschweig, 10.02.2023, Unterschrift

Deutsches Zentrum
DLR für Luft- und Raumfahrt
German Aerospace Center

Detect & Avoid unter der Berücksichtigung von Hindernissen am Beispiel des tieffliegenden unbemannten Luftfahrzeugs "ALAADy-Demonstrator"

Für den Flug unbemannter Luftfahrzeuge außerhalb der Sichtweite des Steuerers werden Detect & Avoid (DAA) Methoden benötigt, die Kollisionen des unbemannten Luftfahrzeuges verhindern. Während entsprechende Systeme bei bemannten Flugzeugen bereits etabliert sind, rückt die Notwendigkeit von DAA Systemen bei unbemannten Luftfahrzeugen immer stärker in den Fokus. Jedoch ist eine Übertragbarkeit der in bemannten Luftfahrzeugen verwendeten Systeme nicht ohne weiteres möglich, da insbesondere für tiefffliegende unbemannte Luftfahrzeuge nicht nur andere Luftverkehrsteilnehmer, sondern beispielsweise auch Luftfahrthindernisse, Geo-Fences oder die Geländehöhe berücksichtigt werden müssen [1].
Im Rahmen der Arbeit soll die DAIDALUS (Detect and AvoID Alerting Logic for Unmanned Systems) [2, 3] Programmbibliothek in das Autopilotensystem des unbemannten Tragschraubers ALAADy-Demonstrators (Automated Low Altitude Air Delivery) [4, 5] integriert und für den Einsatz in niedrigen Flughöhen bis 120m bewertet werden. Zur Erkennung anderer Luftverkehrsteilnehmer wird ein FLARM/ADS-B Empfänger angenommen. Neben der Bewertung der DAA Funktionalität zur Vermeidung von Kollisionen mit anderen Luftverkehrsteilnehmern soll die Möglichkeit zur Berücksichtigung von Hindernissen wie Luftfahrthindernissen oder Geo-Fences betrachtet werden.

Aufgabenstellung im Detail:

- Literaturrecherche zu DAA Methoden und deren Anwendung für tiefffliegende unbemannte Luftfahrzeuge sowie zur Berücksichtigung von Luftfahrthindernissen, Geo-Fences, etc.
- Erstellen eines Konzeptes zum automatischen DAA mithilfe der DAIDALUS Programmbibliothek für den unbemannten Tragschrauber ALAADy-Demonstrator
- Integration der DAIDALUS Programmbibliothek in den Autopiloten des ALAADy-Demonstrators
- Integration des erstellten Konzeptes in die SITL (Software in the Loop) Simulation des ALAADy-Demonstrators
- Bewertung des Ansatzes zum automatischen Ausweichen von anderen Luftverkehrsteilnehmern in niedrigen Flughöhen in der SITL Simulation
- Bewertung des Ansatzes im Hinblick auf die simultane Berücksichtigung von Luftverkehrsteilnehmern und weiteren relevanten Hindernissen, wie z.B. Luftfahrthindernissen, Geo-Fences, etc.
- Dokumentation der Arbeit entsprechend den Richtlinien der TU München

Literatur:

[1] ED-267 Operational Services and Environment Description for Detect and Avoid in Very Low Level Operations, EUROCAE, 2020
[2] https://github.com/nasa/daidalus
[3] DO-365B Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems, RTCA, 2021
[4] https://www.dlr.de/ft/alaady
[5] Softwarearchitektur für einen Unbemannten Luftfrachttransportdemonstrator, S. Benders, L. Goormann, S. Lorenz, & J. C. Dauer, Publikationen zum Deutschen Luft- und Raumfahrtkongress, 2018

Braunschweig, 10.08.2022

Kurzfassung

Soll ein unbemanntes Luftfahrzeug außerhalb der Sichtgrenze eines Steuerers und in einem mit anderen Luftfahrzeugen geteilten Luftraum betrieben werden, wird ein Detect & Avoid (DAA) System benötigt. Derartige Systeme sind in der bemannten Luftfahrt bereits seit Jahren etabliert. Da tieffliegende unbemannte Luftfahrzeuge auf eine Flüghöhe von maximal 120 m über Grund beschränkt sind, ist nur eine begrenzte Ausweichmöglichkeit in vertikaler Richtung gegeben. Des Weiteren bieten die etablierten DAA Systeme nur bedingt Funktionen, um einen sicheren Abstand zu Bauwerken, Geländeerhebungen und Geofences einzuhalten, den tieffliegende unbemannte Luftfahrzeuge wahren müssen. Aufgrund der unterschiedlichen Rahmenbedingungen für bemannte und tieffliegende unbemannte Luftfahrzeuge ist eine Nutzung der etablierten DAA Systeme in tieffliegenden unbemannten Luftfahrzeugen nicht ohne weiteres möglich.

In der vorliegenden Arbeit wird am Beispiel des unbemannten Luftfahrzeug ALAADy-Demonstrators (Automated Low Altitude Air Delivery) untersucht, inwiefern die DAIDALUS (Detect and AvoID Alerting Logic for Unmanned Systems) Programmbibliothek auf tieffliegende unbemannte Luftfahrzeuge übertragen werden kann. Neben dem für DAIDALUS angedachten Anwendungsfall zum Ausweichen gegenüber bewegten Luftfahrthindernissen wird die Funktion von DAIDALUS in dieser Arbeit zusätzlich genutzt, um auch statischen Hindernissen auszuweichen. Im Anwendungsfall des ALAADy-Demonstrators ist die Flughöhe auf maximal 120 m über Grund beschränkt. Daher werden im Rahmen dieser Arbeit Ausweichmanöver über die Änderung des Kurses betrachtet. Es wird geprüft, wie zuverlässig DAIDALUS den jeweiligen Hindernissen ausweicht und wie die Parameter für die Anwendung mit dem ALAADy-Demonstrator optimiert werden können. Die Auswertung der in dieser Arbeit präsentierten Lösung geschieht anhand von Simulationen verschiedener Szenarien. Die betrachteten Szenarien beinhalten sowohl statische als auch bewegte Hindernisse.

Abstract

An unmanned aircraft operated beyond the visual line of sight of the remote pilot needs a detect & avoid (DAA) system. Such systems have already been established in manned aviation for years. As very low level unmanned aircraft are limited to an altitude of 120 m above ground level, there is only limited space for the unmanned aircraft to avoid traffic in vertical directions. Furthermore the DAA systems used in manned aircraft are limited in avoiding buildings, terrain or geofences which have to be avoided by very low level unmanned aircraft. Because of the different circumstances for manned and very low level unmanned aircraft the DAA systems used in manned aircraft are not applicable to very low level unmanned aircraft by default.

In this thesis the unmanned aircraft ALAADy-Demonstrator (Automated Low Altitude Air Delivery) is used to explore if the DAIDALUS (Detect and AvoID Alerting Logic for Unmanned Systems) library can be used for very low level unmanned aircraft. Beside the use case of DAIDALUS to avoid moving obstacles, DAIDALUS is also used to avoid static obstacles in this thesis. The use case of the ALAADy-Demonstrator limits the allowed altitude to 120 m above ground level. Hence this thesis investigates avoidance of obstacles via changing track. It is examined how reliably DAIDALUS avoids the respective obstacles and how the parameters can be optimized for the application with the ALAADy demonstrator. The evaluation of the solution presented in this paper is done by simulating different scenarios. The scenarios considered include both static and moving obstacles.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

A

ACAS - Airborne Collision Avoidance System 11
ADS-B - *Automatic Dependent Surveillance* 4
AGL - *Above Ground Level - über Grund* 1
ALAADy - Automated Low Altitude Air Delivery 1
AP - Autopilot 27
ARC - *Air Risk Category* 6

B

BVLOS - *Beyond Visual Line of Sight* 1

C

C2 - *Control and Command* 4
cFS - core Flight Systems 16
CIC - *Core Interface Computer* 3
CPA - *Closest Point of Approach* 9

D

DAA - *Detect & Avoid* 1
DAEC - Deutscher Aero Club e.V. 20
DAIDALUS - Detect and AvoID Alerting Logic for Unmanned Systems 1
DAL - DAIDALUS-ALAADy 21
DFS - Deutsche Flugsicherung GmbH 7
DLR - Deutsches Zentrum für Luft- und Raumfahrt e.V. 1
DTHR - *Horizontal Distance Threshold* 9
DWD - Deutscher Wetterdienst 19

E

EGPWS - Enhanced Ground Proximity Warning System 15
EUDAAS - *European Detect and Avoid System* 18
EUROCAE - European Organization for Civil Aviation Equipment 6

F

FAA - Federal Aviation Administration 8
FCC - *Flight Control Computer* 3
FMS - *Flight Management System* 20

G

GCS - *Ground Control Station - Bodenkontrollstation* 4
GNSS - *Global Navigation Satellite System* 3
GRC - *Ground Risk Class* 6

I

ICAROUS - Independent Configurable Architecture for Reliable Operations of Unmanned Systems 16
INS - *Inertial Navigation System* 3

K

KI - Künstliche Intelligenz 12

L

LCC - *Logging Control Computer* 3

M

MIDCAS - *Mid-Air Collision Avoidance System* 18
MM - Missionsmanager 2

N

NASA - National Aeronautics and Space Administration - Nationale Luft- und Raumfahrtbehörde (der USA) 1
NED - *North East Down* 31
NIMA - National Imagery and Mapping Agency 19
NMAC - *Near Mid-Air Collision* 8

O

OSM - OpenStreetMap 20

P

PolyCARP - Algorithms and Software for Computations with Polygons 16

R

RA - *Resolution Advisory* 11
RL - *Robustness Level* 6
RPAS - *Remotely Piloted Aircraft System* 18
RTCA - Radio Technical Commission for Aeronautics 1

S

SAIL - *Specific Assurance and Integrity Level* 6
SITL - *Software In The Loop* 33
SORA - *Specific Operational Risk Assesment* 6
SRTM - Shuttle Radar Topography Mission 19

T

TA - *Traffic Awareness* 11
TCAS - Traffic Alert and Collision Avoidance System 11

U

V

W

Z

Symbolverzeichnis

χ	Bahnazimutwinkel
γ	Bahnneigungswinkel
ϕ	Hängewinkel
χ	Kurs
B	Breite
d	Distanz
d_0	Anfangsdistanz
f	Updaterate
Δh	Höhendifferenz
$r_{w,min}$	minimaler Wenderadius
Δt	Zeitdifferenz
T	Lookahead Time
v_A	Fluggeschwindigkeit
u	Geschwindigkeit entlang der x-Achse
v	Geschwindigkeit entlang der y-Achse
w	Geschwindigkeit entlang der z-Achse
$\vec{x}$	Positionsvektor
x_g	x-Position im erdlotfesten kartesischen Koordinatensystem
y_g	y-Position im erdlotfesten kartesischen Koordinatensystem
z_g	z-Position im erdlotfesten kartesischen Koordinatensystem
Δy	Differenz in y-Richtung
$DTHR$	Horizontal Distance Threshold
$ZTHR$	Vertical Distance Threshold
$TTHR$	Time Threshold
$TCOA$	Time to Co-Altitude

1. Einleitung

1.1. Motivation

Das Potential von unbemannten Luftfahrzeugen (engl. *Unmanned Aircraft* (UA)) wird mit einem steigenden Grad der Automatisierung und dem Betrieb außerhalb der Sichtweite (engl. *Beyond Visual Line of Sight* (BVLOS)) eines Steuerers in Zukunft weiter ausgeschöpft. Für ein vollständig automatisiertes UA wird eine Funktion zum automatischen Ausweichen von statischen Hindernissen und anderen Luftverkehrsteilnehmern benötigt. Statische Hindernisse können zum Beispiel gesperrte Lufträume oder Bauwerke sein. Andere Luftverkehrsteilnehmer treten zum Beispiel als Segelflugzeuge, motorisierte Flugzeuge, Hubschrauber aber auch in der Form von Ballonen, Hängegleitern und anderen UA auf. Für tieffliegende (engl. *Very Low Level* (VLL)) UA ist speziell der Luftraum bis $120\,\mathrm{m\,AGL}$ (*Above Ground Level - über Grund*) vorgesehen [33]. In Höhen bis $120\,\mathrm{m\,AGL}$ gibt es statische Hindernisse wie zum Beispiel Gebäude, Stromleitungen aber auch Geländeerhöhungen, welche zuverlässig umflogen werden müssen. In dieser Arbeit wird eine Ausweichfunktion für das vom Deutschen Zentrum für Luft- und Raumfahrt e.V. (DLR) entwickelte und betriebene UA *ALAADy-Demonstrator* betrachtet. ALAADy steht für *Automated Low Altitude Air Delivery*. Die im DLR unter dem Akronym angesiedelten Projekte dienen der Erforschung des automatisierten Fluges und Frachttransports in niedrigen Höhen bis $120\,\mathrm{m\,AGL}$ [50]. Beim ALAADy-Demonstrator handelt es sich um einen ursprünglich manntragenden Tragschrauber, der vom DLR als Demonstrator für den automatisierten Flug von UA umgerüstet wurde. Eine genauere Beschreibung des ALAADy-Demonstrators ist in Unterkapitel 1.3 zu finden.
In Zukunft soll der ALAADy-Demonstrator auch außerhalb der Sichtgrenze automatisiert geflogen werden. Für den sicheren automatisierten Betrieb des UA ist es nötig, anderen Luftfahrzeugen und Hindernissen auszuweichen. Die von der amerikanischen Luft- und Raumfahrtbehörde NASA entwickelte Detect and AvoID Alerting Logic for Unmanned Systems (DAIDALUS) Programmbibliothek [58] ermöglicht die Implementierung eines *Detect & Avoid* (DAA) Algorithmus und ist eine Referenzimplementierung des von der Radio Technical Commission for Aeronautics (RTCA) erarbeiteten Standards DO-365A [43]. DAIDALUS wird von der NASA unter der Open Source Lizenz „NASA's Open Source Agreement“ bereitgestellt [58].
DAIDALUS wird im Rahmen dieser Arbeit in den bestehenden Autopiloten des ALAADy-Demonstrators integriert. Dazu wird eine Kommunikationsschnittstelle zwischen beiden Systemen erstellt und DAIDALUS innerhalb der Flugleistungsgrenzen des ALAADy-Demonstrators betrieben. Durch eine Simulation wird unter Verwendung eines kinematischen Bewegungsmodells die Zuverlässigkeit von DAIDALUS gegenüber verschiedenen Hindernissen im Anwendungsfall des ALAADy-Demonstrators geprüft. Die Ergebnisse dieser Simulation werden in Einzelfällen durch eine zweite Simulation, die ein den ALAADy-Demonstrator nachbildendes dynamisches Bewegungsmodell nutzt, validiert.

1.2. Aufbau der Arbeit

In diesem Kapitel 1 wird die Motivation hinter dieser Arbeit sowie ihr Aufbau dargestellt. Des Weiteren wird der ALAADy-Demonstrator vorgestellt. In Kapitel 2 Stand der Technik wird eine Auswahl der beim Betrieb von tiefliegenden UA einzuhaltenden Standards und Regularien vorgestellt, bevor eine Einführung in DAA Systeme gegeben wird. Der Einblick in bestehende Techniken von DAA Systemen wird in Unterkapitel 2.2.2 vertieft. Abschließend werden in Unterkapitel 2.3 mögliche (quasi-)statische Hindernisse sowie mögliche Datenquellen dazu vorgestellt. In Kapitel 3 wird die Funktionsweise von DAIDALUS näher erörtert und die im Rahmen dieser Arbeit erstellte und genutzte Anwendung der DAIDALUS Programmbibliothek vorgestellt. Unterkapitel 3.2 gibt einen Überblick über den Missionsmanager (MM) des ALAADy-Demonstrators, die im Rahmen dieser Arbeit erstellte Schnittstelle zwischen ALAADy-Demonstrator und DAIDALUS sowie die im Rahmen dieser Arbeit getroffenen Annahmen. Abschließend werden die zur Erarbeitung der präsentierten Ergebnisse verwendeten Simulationsumgebungen vorgestellt. Anhand repräsentativer Szenarien wird die Nutzung der DAIDALUS Programmbibliothek zum Ausweichen statischer Hindernisse in Kapitel 4 gezeigt. Des Weiteren werden die erarbeiteten Ergebnisse zu Szenarien mit bewegten Hindernissen und beiden Arten von Hindernissen präsentiert und diskutiert. Eine Zusammenfassung der gewonnenen Erkenntnisse ist in Kapitel 5 zu finden. Abschließend gibt Kapitel 6 einen Ausblick auf mögliche weitere Untersuchungen und nächste Schritte zur Anwendung der DAIDALUS Programmbibliothek für VLL UA.

1.3. ALAADy-Demonstrator

Das für diese Arbeit betrachtete unbemannte Luftfahrzeug ist der ALAADy-Demonstrator des Instituts für Flugsystemtechnik des DLR in Braunschweig. Abbildung 1.1 zeigt den ALAADy-Demonstrator.

Abbildung 1.1 ALAADy-Demonstrator [48]

Der Demonstrator basiert auf einem Tragschrauber des Typs AutoGyro MTOfree und wurde vom Institut für Flugsystemtechnik für den unbemannten und automatisierten Flug modifiziert [12]. Erste Flugkampagnen mit dem ALAADy-Demonstrator wurden bereits erfolgreich durchgeführt [13]. Abbildung 1.2 zeigt eine vereinfachte Darstellung des bordseitigen Avioniksystems des ALAADy-Demonstrators.

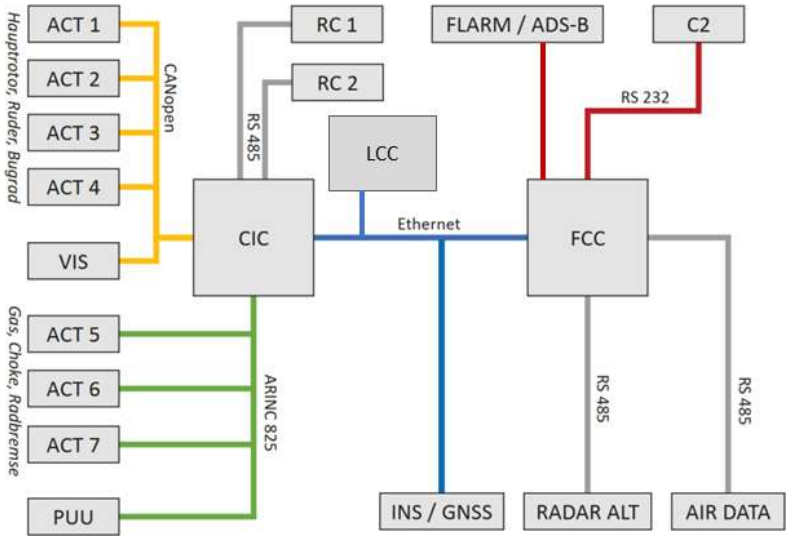


Abbildung 1.2 Vereinfachte Darstellung des ALAADy-Avioniksystems in Anlehnung an [12]

Der ALAADy-Demonstrator besitzt einen *Core Interface Computer* (CIC) zum Ansteuern der Aktuatoren. Der CIC ist mit dem *Logging Control Computer* (LCC), dem *Flight Control Computer* (FCC) und Sensoren zur Flugzustandserfassung verbunden. Die Daten werden dabei über ein *Global Navigation Satellite System* (GNSS) gestütztes *Inertial Navigation System* (INS), den Nasenmast und einen Radar-Höhenmesser gewonnen. Der LCC dient der Datenaufzeichnung im Betrieb. Der FCC dient zur Berechnung der Flugbahn, der Ausführung von

experimenteller Software sowie zur Kommunikation über Datenlink (*Control and Command* (C2)) mit der Bodenstation (engl. *Ground Control Station - Bodenkontrollstation* (GCS)). Der FCC ist mit mehreren Sensoren verbunden. [12] Der Autopilot des ALAADy-Demonstrators wird auf dem FCC ausgeführt. Durch Vorgabe der Fluggeschwindigkeit v_A , des Bahnneigungswinkels γ und des Bahnazimutwinkels χ kann die Flugbahn des Luftfahrzeugs durch ein Programmmodul wie zum Beispiel die Integration von DAIDALUS kommandiert werden.
Die im Rahmen dieser Arbeit erstellte Integration des DAIDALUS Algorithmus wird später auf dem FCC ausgeführt werden. Die für DAIDALUS nötigen FLARM und ADS-B Daten anderer Luftfahrzeuge werden dem FCC von einem PowerFLARM Core ADS-B Modul [6] bereitgestellt. Das Modul übergibt Informationen zur Position von anderen detektierten Luftfahrzeugen, sowie deren Vertikal- und Horizontalgeschwindigkeit. Darüberhinaus werden jeweils Informationen zum Typ der detektierten Luftfahrzeuge, eine Identifikationsnummer und das von den Luftfahrzeugen ausgehende Alarmlevel ausgegeben. Die Daten des detektierten Luftfahrzeugs mit der geringsten Distanz werden mit einer Frequenz von $1\,\mathrm{Hz}$, die Daten der anderen empfangenen Luftfahrzeuge werden in unregelmäßigen Abständen ausgegeben. [4] Die Positions- und Geschwindigkeitsdaten des ALAADy-Demonstrators werden dem FCC durch das INS zur Verfügung gestellt. Eine genauere Beschreibung des auf dem FCC ausgeführten und für diese Arbeit relevanten MissionsManager des ALAADy-Demonstrators ist in Abschnitt 3.2.1 gegeben.

2. Stand der Technik

In diesem Kapitel wird ein Überblick über die Standards und Regularien zum Betrieb von unbemannten Luftfahrzeugen (Abschnitt 2.1.1) sowie der aktuelle Stand der Technik im Bereich von DAA Systemen von UA mit dem Fokus auf VLL UA (Abschnitt 2.1.2) aufgezeigt. In Unterkapitel 2.3 wird ein Überblick über mögliche (quasi-)statische Hindernisse und den dazugehörigen Datenquellen gegeben.

2.1. Standards und Regularien

In diesem Unterkapitel wird ein Überblick über die einzuhaltenden Standards und Regularien in Bezug auf DAA von UA gegeben. Dabei werden zuerst allgemeine Regularien betrachtet, die beim Führen von bemannten wie auch unbemannten Luftfahrzeugen beachtet werden müssen. Im anschließenden Abschnitt wird eine Auswahl von Standards und Regularien vorgestellt, die den Betrieb eines UAs beschreiben.

2.1.1. Ausweichregeln in der Luftfahrt

Der Pilot eines Luftfahrzeugs ist verpflichtet, anderen Luftfahrzeugen auszuweichen. Dabei gelten die in den Rules of the Air [41] festgehaltenen Ausweichregeln:

- Sind zwei Luftfahrzeuge auf (nahezu) frontalem Kollisionskurs, so haben beide Luftfahrzeuge nach rechts auszuweichen.
- Ein Luftfahrzeug hat ein anderes Luftfahrzeug auf der rechten Seite zu überholen.
- Ein von rechts kommendes Luftfahrzeug hat einem von links kommenden Luftfahrzeug nach rechts auszuweichen.

Dabei gilt immer, dass...

- ... motorbetriebene Luftfahrzeuge Luftschiffen, Segelflugzeugen (inkl. Hängegleitern) und Ballonen auszuweichen haben [41].
- ... Luftschiffe Segelflugzeugen (inkl. Hängegleitern) und Ballonen auszuweichen haben [41].
- ... Segelflugzeuge (inkl. Hängegleiter) Ballonen auszuweichen haben [41].
- ... motorbetriebene Luftfahrzeugen anderen Luftfahrzeugen im Schleppbetrieb auszuweichen haben [41].
- ... unbemannte Luftfahrzeuge gegenüber bemannten Luftfahrzeugen immer ausweichpflichtig sind [66].

Abgesehen von Hubschraubern, motorbetriebenen Fallschirmen und Gewicht-gesteuerten Luftfahrzeugen haben alle (bemannten) Luftfahrzeuge eine Sicherheitshöhe von mindestens 500 ft AGL[1] einzuhalten [41]. Aufgrund dieser Sicherheitshöhe entstand die Idee, tiefffliegende UA (bis $120\,\mathrm{m\,AGL}$) einzusetzen und so den Großteil der bemannten Luftfahrzeuge zu meiden [42, 2]. Dennoch müssen auch tiefffliegende UA auf möglichen Verkehr achten und diesem ausweichen. Der Verkehr kann beispielsweise in Form von Rettungshubschraubern, Segelflugzeugen oder anderen UA auftreten.

2.1.2. Ausweichregeln für unbemannte Luftfahrzeuge

Mit der Durchführungsverordnung (EU) 2019/947 wurden für die EU die Klassen *open (offen)*, *specific (speziell)* und *certified (zulassungspflichtig)* für UA definiert. Die Einordnung in die einzelnen Klassen geschieht auf Basis des Betriebsrisikos des jeweiligen UAs. Das Betriebsrisiko wird sowohl vom UA selbst, als auch vom Einsatzzweck beeinflusst. Beispielsweise hat die charakteristische Größe des UAs einen unmittelbaren Einfluss auf die Einordnung. UA mit geringem Betriebsrisiko werden in die Klasse *offen* eingeordnet. UA mit hohem Betriebsisiko werden in die Klasse *zulassungspflichtig* eingeordnet. Dazwischen ist die Klasse *speziell* definiert. [19]

Zur Klasse *offen* gehören im Allgemeinen privat gebaute UA – also zum Beispiel Modellflugzeuge und -Hubschrauber - mit einer maximalen Startmasse von unter 25kg. Diese werden durch den Fernpiloten immer im Sichtbereich (engl. *Visual Line of Sight* (VLOS)) und unter $120\,\mathrm{m\,AGL}$ kontrolliert. Menschenansammlungen dürfen nicht überflogen werden. [19]

UA werden der Klasse *speziell* zugeordnet, wenn sie eine maximale Startmasse von 25 kg überschreiten, oberhalb von $120\,\mathrm{m\,AGL}$, in speziellen Lufträumen oder außerhalb des Sichtbereichs betrieben werden. [51, 19]

UA werden der Klasse *zulassungspflichtig* zugeordnet, wenn sie über Menschenansammlungen betrieben werden sollen, oder sie zum Transport von Gefahrgut oder Menschen konstruiert sind [51, 19].

Das für diese Arbeit betrachtete UA ALAADy-Demonstrator besitzt eine Betriebsgenehmigung in der Klasse *speziell* mit *Specific Assurance and Integrity Level* (SAIL) II. Eine Betriebsgenehmigung mit SAIL III soll zeitnah realisiert werden. SAIL bezeichnet eine Unterkategorie innerhalb der Klasse *speziell*. Damit wird eine detailiertere Risikoeinordnung gewährleistet.

Die von der European Organization for Civil Aviation Equipment (EUROCAE) veröffentlichte *Operational Services & Environment Definition (OSED) for Detect & Avoid in Very Low-Level Operations ED-267* [33] beschreibt den *Specific Operational Risk Assesment* (SORA) Prozess, der die Grundlage für die Einordnung von UA der Klasse *speziell* entsprechend der jeweiligen SAIL Level bietet. In diese Einordnung fließt neben dem geplanten Verwendungszweck des UAs auch die dabei erwartete *Ground Risk Class* (GRC), die erwartete *Air Risk Category* (ARC) sowie die *Tactical Mitigation Performance Requirements* (TMPR) und das *Robustness Level* (RL) ein. Die ARC kann durch die Verwendung von Tactical Mitigations reduziert werden. Ein DAA System gilt hierbei als Tactical Mitigation [33]. Die höchsten TMPR in den ED-267 [33] fordern für ein DAA System die Einhaltung der von EUROCAE WG-105

[1] ca. 150 m über Grund

oder RTCA-228 erarbeiteten Standards. Da DAIDALUS eine Referenzimplementierung der von der RTCA-228 erarbeiteten DO-365A [43] ist, bietet die Nutzung von DAIDALUS als DAA System eine gute Voraussetzung für eine Reduzierung der ARC. Dies kann zu einer geringeren SAIL-Einstufung für das UA mit DAA System führen.
Ein möglicher Aufbau eines DAA Systems, das den geforderten TMPR entspricht, ist in den ED-267 [33] veröffentlicht. Hierin werden eine Vielzahl unterschiedlicher Szenarien und Umgebungen für den Betrieb von VLL UA betrachtet. Neben dem Betrieb von VLL UA im urbanen und suburbanen Bereich werden vor allem die sogenannten *Very Low Level Air Traffic Management* (VLLATM) Services und die Notwendigkeit von DAA Systemen zum Ausweichen gegenüber anderen bemannten und unbemannten Luftfahrzeugen, Terrain, mobilen und statischen Hindernissen sowie gefährlichem Wetter und Lebewesen betrachtet. Als *Very Low-Level* wird in den ED-267 der Bereich definiert, in dem keine bemannten Luftfahrzeuge zu erwarten sind. Im Rahmen der ED-267 [33] ist das im Falle von bebautem Gebiet oder über Menschenansammlungen der Bereich bis $300\,\mathrm{m}$ über dem höchsten Hindernis in einem Radius von $600\,\mathrm{m}$. Andernfalls wird eine Höhe von $150\,\mathrm{m}$ über Grund beziehungsweise über dem höchsten Hindernis innerhalb eines Radius von $150\,\mathrm{m}$ als Grenze des VLL definiert.
Um das Potential von VLL UA umfänglich nutzen zu können, ist ein sicherer Betrieb BVLOS unumgänglich [33]. Dafür müssen dem Fernpiloten aktuelle Daten zu möglichen Gefahren vorliegen. Zu diesen Gefahren gehören laut den ED-267 [33] unter anderem:

- andere Luftfahrzeuge
- Gelände und Hindernisse auf dem Boden (inklusive Fahrzeuge)
- gefährliche Wetterbedingungen
- Menschen und Tiere

Die Daten für diese Hindernisse stammen in der praktischen Umsetzung je nach Hindernis aus verschiedenen Quellen. Dazu zählen sowohl Datenbanken für Gelände und statische Hindernisse, sowie verschiedene Sensoren und Detektoren für mobile oder sich bewegende Hindernisse.

2.2. Detect & Avoid

Obwohl der Luftraum bis $500\,\mathrm{ft\,AGL}$ deutlich weniger frequentiert ist als die kontrollierten Lufträume, muss ein UA in der Lage sein, anderen Luftfahrzeugen oder Hindernissen selbstständig auszuweichen. Es ist keine Quelle bekannt, die die Anzahl der Luftfahrzeuge im unkontrollierten Luftraum oder in Höhen bis $500\,\mathrm{ft\,AGL}$ aufzeigt. Die geringe Datenlage ist vor allem damit zu begründen, dass in Deutschland im Luftraum G nach Sichtflugregeln (engl. *Visual Flight Rules* (VFR)) geflogen wird und somit keine Transponderpflicht besteht [31]. Luftfahrzeuge ohne Transponder sind statistisch (quasi) nicht zu erfassen. Allerdings veröffentlicht die Deutsche Flugsicherung GmbH (DFS) jährlich die Anzahl der VFR Events (Starts, Landungen sowie tiefe Überflüge und Touch'n gos) auf deutschen (internationalen) Verkehrs-

flughäfen. Demnach gab es im Jahr 2020 insgesamt 93.762 und im Jahr 2021 insgesamt 102.032 solcher VFR Events an deutschen (internationalen) Verkehrsflughäfen [37]. Zwar werden die vielen kleinen Flugplätze in Deutschland dabei nicht beachtet. Die Anzahl der von dort gestarteten (und in der Statistik nicht aufgeführten) Flüge, wird aber als relevant eingeschätzt.
Im Folgenden gibt Abschnitt 2.2.1 eine Einführung in die Funktionsweise und Begrifflichkeit von DAA Systemen im Allgemeinen. In Abschnitt 2.2.2 werden mögliche Datenquellen und Sensoren zum Erkennen von anderen Luftfahrzeugen und Hindernissen präsentiert. Abschnitt 2.2.3 beschäftigt sich mit Algorithmen zur Berechnung des Ausweichkurses für DAA Systeme. Die Abschnitte 2.2.4 und 2.2.5 stellen die gängigen DAA Systeme für bemannte beziehungsweise unbemannte Luftfahrzeuge vor. Abschnitt 2.2.6 gibt einen kurzen Vergleich zwischen den Fähigkeiten von Systemen für bemannte und unbemannte Luftfahrzeuge.

2.2.1. Einführung

Zur Kategorisierung der einzelnen Zustände während DAA Szenarien wurden bereits 2009 von der Federal Aviation Administration (FAA) so genannte *Separation Volumes* definiert [16]. Auch die DO-365 [42] nutzen diese Volumen zur Einordnung der jeweiligen Zustände. Der Mittelpunkt der zylindrischen Volumen ist mit der Position des jeweiligen Luftfahrzeugs definiert. In den DO-365 [42] werden die Volumen als *Preventive Volume*, *Corrective Volume*, *Recovery Volume* und *NMAC*[2] *Volume* bezeichnet. Die Volumen sind ineinander verschachtelt und werden von *Preventive* zu *NMAC* kleiner. Die Volumen können entweder um das eigene Luftfahrzeug (Ownship) oder jeweils um die Hindernisse aufgespannt werden. Im Rahmen dieser Arbeit werden die Volumen immer um den Mittelpunkt der Hindernisse aufgespannt. Im folgenden Absatz wird im Sinne der Verständlichkeit lediglich von einem Luftfahrzeug als Hindernis gesprochen. Die Gleichungen sowie die Beschreibungen sind auf ein statisches Hindernis übertragbar.
Die Abmessungen der genannten Volumen werden (mit Ausnahme des *NMAC Volumes*) in ihrer vertikalen und horizontalen Ausprägung jeweils durch eine Zeit- und eine Längenkonstante definiert. Wird ein Volumen eines Hindernisses verletzt, so spricht man von einer *Well-Clear Violation (WCV)*. Dabei gilt ein Volumen als Verletzt, sofern die folgenden Bedingungen erfüllt sind:

$$WCV \equiv HWCV \wedge VWCV \tag{2.1}$$

Dabei steht $HWCV$ für eine *WCV* in der horizontalen Ebene und $VWCV$ für eine *WCV* in vertikaler Richtung. $HWCV$ und $VWCV$ sind wie folgt definiert:

$$HWCV \equiv d \leq DTHR \vee (HMDF \wedge 0 \leq \tau_{mod} \leq TTHR) \tag{2.2}$$

$$VWCV \equiv \Delta h \leq ZTHR \vee 0 \leq t_{coa} \leq TCOA \tag{2.3}$$

Die Größe d beschreibt die horizontale Distanz zum Hindernis. $DTHR$ und $TTHR$ entsprechen der Längen- bzw. Zeitkonstante des jeweiligen Volumens in horizontaler Richtung. τ_{mod} ist die Zeit, die das Ownship benötigt, bis das Volumen in horizontaler Richtung erreicht wird.

[2] Near Mid-Air Collision (beinahe Kollision in der Luft)

$ZTHR$ und $TCOA$ sind die Längen- bzw. Zeitkonstante des jeweiligen Volumens in vertikaler Richtung. Δh beschreibt die Höhendifferenz der beiden Luftfahrzeuge in einer Längeneinheit, t_{coa} gibt die Zeit, in der beide Luftfahrzeuge auf der gleichen Höhe sind, an. $HMDF$ steht für *Horizontal Miss-Distance Filter* und ist als $HMDF \equiv d_{coa} \leq HMD$ definiert. Die Distanzfunktion d_{coa} beschreibt die projizierte horizontale Distanz der beiden Luftfahrzeuge an ihrem *Closest Point of Approach* (CPA) also dem Punkt, an dem sich beide Luftfahrzeuge bei Beibehaltung der aktuellen Geschwindigkeitsvektoren am nächsten sind. HMD bezeichnet die *Horizontal Miss-Distance* (horizontaler Abstand, um den sich beide Luftfahrzeuge verfehlen) und wird üblicherweise auf denselben Wert wie $DTHR$ gesetzt. [55, 29] Die in den DO-365 [42] definierten Werte für die oben genannten Parameter sind in Tabelle 2.1 aufgeführt.

Tabelle 2.1 Parameter der *Separation Volumes* gemäß DO-365 [42]

Parameter	*Preventive Volume*	*Corrective Volume*	*Recovery Volume*	*NMAC Volume*
DTHR [nmi]	0,66[3]	0,66[3]	0,66[3]	500[m]
ZTHR [ft]	700[4]	450[5]	450[5]	20[m]
TTHR [s]	35	35	35	-
TCOA [s]	0	0	0	-

Die Längenkonstante der Volumen wird in horizontaler Ausprägung als *Horizontal Distance Threshold (DTHR)*, in vertikaler Ausprägung als *Vertical Distance Threshold (ZTHR)* bezeichnet. Die Zeitkonstante wird in horizontaler Ausprägung *Time Threshold (TTHR)* beziehungsweise τ^*_{mod} genannt. In vertikaler Ausprägung spricht man von der *Time to Co-Altitude (TCOA)*. Die Bezeichnungen *DTHR*, *ZTHR*, *TTHR* und *TCOA* werden in der Konfigurationsdatei für DAIDALUS gleich genannt.
Abbildung 2.1 zeigt einen Größenvergleich der ineinander verschachtelten Volumen nach Definition der DO-365. Im oberen Teil ist eine Draufsicht auf das Luftfahrzeug und seine Hindernisse gegeben. Der untere Teil der Abbildung zeigt die Seitenansicht.

[3] ca. 1220 m
[4] ca. 215 m
[5] ca. 120 m

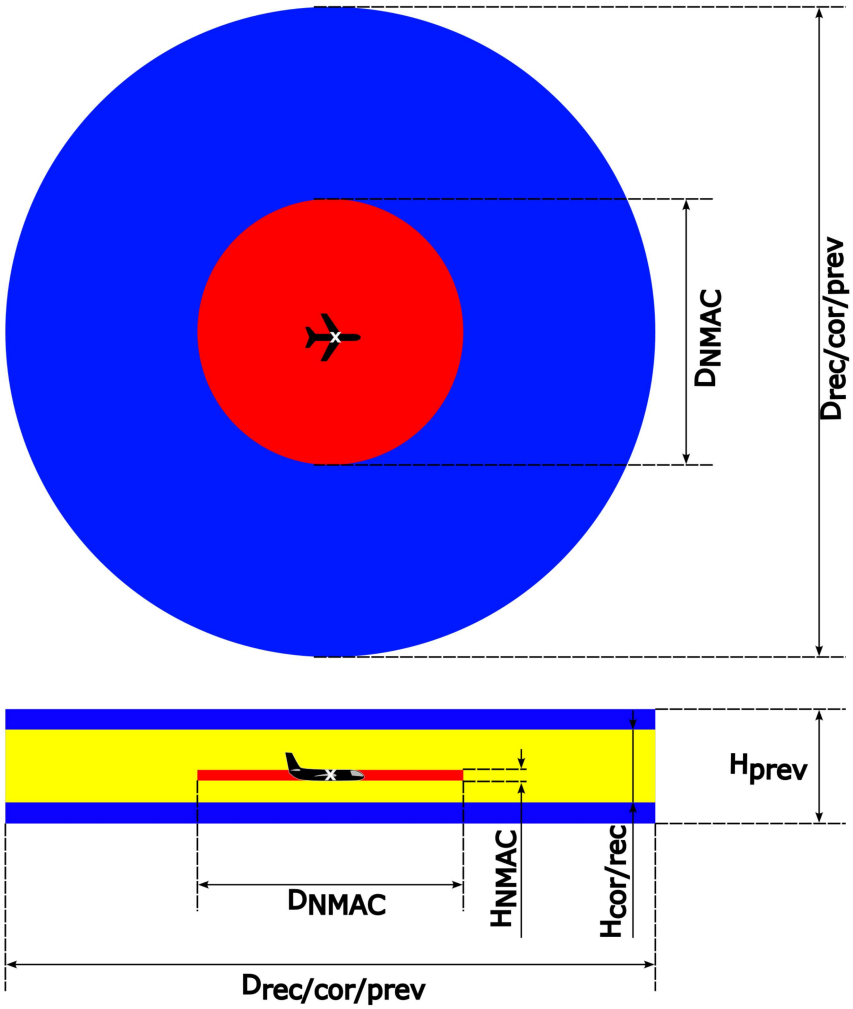


Abbildung 2.1 Größenvergleich der zylindrischen Volumen eines DAA Systems gemäß den DO-365 [42]

Wie der Tabelle und der Abbildung zu entnehmen ist, sind die Volumen *Preventive Volume*, *Corrective Volume* und *Recovery Volume* in ihrer horizontalen Ausprägung identisch. Sie sind in Abbildung 2.1 in der Draufsicht als blauer Kreis dargestellt. In der Seitenansicht ist das *Preventive Volume* in blauer Farbe dargestellt. Das *Corrective Volume* ist in der vertikalen Ausprägung ebenfalls deckungsgleich mit dem *Recovery Volume* und ist in der Seitenansicht in gelber Farbe abgebildet. Das *NMAC Volume* ist sowohl in der Draufsicht als auch in der Seitenansicht durch die rote Farbe gekennzeichnet. Die Proportionen der Höhen sind im Vergleich zu den Durchmessern richtig dargestellt. Die in der Abbildung angegebenen Bezeichnungen D_{NMAC} und $D_{rec/cor/prev}$ bezeichnen die Durchmesser der jeweiligen Volumen. Die in Tabelle 2.1 angegebenen Längenangaben bezeichnen dagegen den Radius und die halbe Höhe der Volumen. Es gilt also $D_{Volumen} = 2 \cdot DTHR_{Volumen}$ und $H_{Volumen} = 2 \cdot ZTHR_{Volumen}$.

Neben den oben dargestellten Längenkonstanten ist die Zeitkonstante τ_{mod} in den DO-365 [42] wie folgt definiert:

$$\tau_{mod} = \frac{-(r^2 - DMOD^2)}{r \cdot \dot{r}} \tag{2.4}$$

Die horizontale Distanz r ist die Distanz zwischen den beiden zu betrachtenden Luftfahrzeugen. Die Distanz reicht allerdings nicht vom Mittelpunkt des Ownships zum Mittelpunkt des fremden Luftfahrzeugs, sondern ist um die Längenkonstante $DTHR$ gekürzt. Daraus folgt $r = d - DTHR$. Die horizontale Änderung von r wird mittels $\dot{r}$ ausgedrückt. Eine Abnahme der horizontalen Distanz wird durch einen positiven Wert von $\dot{r}$ beschrieben. $DMOD$ wird die Distanzmodifikation von τ_{mod} genannt und besitzt üblicherweise denselben Wert wie $DTHR$. $TTHR$ entspricht dem Grenzwert τ^*_{mod}, bei deren Unterschreitung eine Verletzung des jeweiligen Volumens vorliegt.
Sofern keines der genannten Volumen verletzt wird, spricht man im Englischen von *remain well clear* (deutsch etwa: genügend Abstand einhalten). Bei Verletzung des *Preventive Volumes* durch ein fremdes Luftfahrzeug soll lediglich auf dieses Luftfahrzeug aufmerksam gemacht werden (engl. *Traffic Awareness* (TA)). Bei Verletzung des *Corrective Volumes* soll eine Kursänderung des eigenen Luftfahrzeugs stattfinden, um dem fremden Luftfahrzeug mit ausreichend Abstand auszuweichen und wieder in den Bereich des well clear zu gelangen (engl. maintain well clear) [61]. Geschieht die Anpassung des Kurses nicht rechtzeitig oder ist die Anpassung zu gering, kommt es zur Verletzung des *Recovery Volumes*. Bei Verletzung des *Recovery Volumes* muss einer Ausweichempfehlung gefolgt werden, um den Konflikt aufzulösen und die *NMAC* zu vermeiden. Eine *NMAC* ist als Verletzung des *NMAC Volumes* definiert. Die Ausweichempfehlung, die zur Auflösung des Konflikts führt bezeichnet man im Englischen als *Resolution Advisory (RA)*. [61]
In der Literatur wird ein fremdes Luftfahrzeug - abhängig von der Distanz zum Ownship - als *Traffic*, *Intruder* oder *Threat* bezeichnet. Die genauen Distanzen, die die jeweilige Bezeichnung definieren, sind je nach Quelle verschieden. Dabei ist das als Traffic bezeichnete Luftfahrzeug am weitesten entfernt, das als Threat bezeichnete Luftfahrzeug ist dem Ownship am nächsten. Im Rahmen dieser Arbeit wird ein fremdes Luftfahrzeug als *Traffic (Aircraft)* bezeichnet, wenn es sich außerhalb des *Corrective Volumes* (aber gegebenenfalls innerhalb des *Preventive Volumes*) befindet. Ein Luftfahrzeug, das das *Corrective Volume* des Ownships verletzt, wird als Intruder bezeichnet [42]. Wird zusätzlich das *Recovery Volume* verletzt, so wird dieses Luftfahrzeug im Rahmen dieser Arbeit als *Threat* bezeichnet.

2.2.2. Detect

Für das Erkennen (Detect) von anderen Luftfahrzeugen gibt es verschiedene Möglichkeiten. Eine Möglichkeit der Erkennung von anderen Luftfahrzeugen besteht darin, Datensätze von diesen zu empfangen. Die Datensätze beinhalten zum Beispiel Informationen zur aktuellen Position, Höhe, der Richtung und Geschwindigkeit des jeweiligen Luftfahrzeugs und werden je nach Klasse des Luftfahrzeugs über unterschiedliche Protokolle geteilt. Die nachfolgend aufgeführten Mode-S, ADS-B Transpondern und FLARM Module sind jeweils als kooperative Systeme aufgebaut. Die Transponder und Module kommunizieren mit dem jeweiligen Protokoll miteinander und tauschen gegenseitig Informationen aus.
In Europa ist für Luftfahrzeuge mit mindestens 19 Sitzplätzen oder einer maximalen Startmasse von $5,7\,\mathrm{t}$ ein Airborne Collision Avoidance System (ACAS)/Traffic Alert and Collision Avoidance System (TCAS) verpflichtend [31]. TCAS verwendet zur Erkennung von anderen

Luftfahrzeugen den im Luftfahrzeug verbauten Mode-S Transponder [35]. Seit dem 7. Dezember 2020 ist außerdem ein ADS-B Transponder für alle Luftfahrzeuge mit einer maximalen Startmasse von mindestens $5,7\,\mathrm{t}$ oder einer Reisegeschwindigkeit von mindestens $250\,\mathrm{kts}$[6] verpflichtend. Luftfahrzeuge, die eine Startmasse von mindestens $5,7\,\mathrm{t}$ oder eine Reisegeschwindigkeit von mindestens $250\,\mathrm{kts}$ aufweisen, mit erstem Lufttüchtigkeitszeugnis vor dem 7. Dezember 2020 müssen bis zum 7. Juni 2023 mit einem ADS-B Transponder ausgerüstet sein [18]. ADS-B spielt in den DO-365 Standards [42] eine wesentliche Rolle für die zuverlässige Nutzung eines DAA Systems. Bezogen auf den Anwendungsfall in dieser Arbeit - den tieffliegenden ALAADy-Demonstrator - gilt es jedoch zu beachten, dass die DO-365 Standards [42] nicht für tieffliegende Luftfahrzeuge erstellt wurden. Luftfahrzeuge, die über einen ADS-B Transponder verfügen müssen, fliegen in der Regel in einer Höhe über $500\,\mathrm{ft\,AGL}$. Dadurch ist die ADS-B Funktion für den ALAADy-Demonstrator nicht zwingend erforderlich, kann aber eine gute Ergänzung darstellen.
Seit einigen Jahren hat sich für Luftsportgeräte wie Hängegleiter und Segelflugzeuge und leichte Luftfahrzeuge FLARM als Standard etabliert. Auch Hubschrauber und Luftfahrzeuge der allgemeinen Luftfahrt sind oft mit einem FLARM Modul ausgestattet. Diese Fluggeräte machen den Großteil der zu erwartenden bewegten Hindernissen des ALAADy-Demonstrators aus. Ein FLARM Modul bietet im Anwendungsfall des ALAADy-Demonstrators daher eine wichtige Grundlage zur Erkennung von fremden Luftfahrzeugen. [3]
Neben den oben beschriebenen kooperativen Erkennungssystemen werden auch nicht kooperative Sensoren eingesetzt. Beispielsweise werden elektro-optische Systeme (z.B. Kameras) verwendet, um mittels künstlicher Intelligenz (KI) mögliche Hindernisse zu erkennen [64]. Diese Art der *Traffic*-Erkennung ist sehr rechenintensiv und benötigt zum Teil eine große Anzahl (unterschiedlicher) Bilddaten, um das System anzulernen und so eine zuverlässige *Traffic*-Erkennung zu gewährleisten [64]. Sofern das System ausreichend trainiert wurde ermöglicht ein Kamera basiertes System eine selbstständige Erkennung von Luftfahrzeugen und ist nicht auf externe Signale von Transpondern angewiesen. Weitere sensorbasierte Möglichkeiten zur Ortung von anderen Luftfahrzeugen basieren auf der Verwendung von RADAR-, LIDAR-, SONAR- oder akustischen Sensoren.

2.2.3. Avoid

Nach der Erkennung des Intruders wird üblicherweise dessen Position und Geschwindigkeitsvektor ausgewertet und anschließend beurteilt, ob ein Manöver zum Ausweichen (Avoid) erforderlich ist. Zur Bewertung einer möglichen Kollision und zur Berechnung eines passenden Ausweichmanövers gibt es folgende Ansätze.
Ein geometrischer Ansatz [10] betrachtet den Vektor der Relativgeschwindigkeit zwischen dem eigenen Luftfahrzeug und einem Intruder. Dder Geschwindigkeitsvektor des Intruders wird als konstant angenommen. Zeigt dieser Vektor der Relativgeschwindigkeit vom Ownship auf den Intruder, so besteht Kollisionsgefahr. Je geringer die Distanz zwischen den beiden Luftfahrzeugen oder je höher die Relativgeschwindigkeit dabei ist, desto dringender ist diese

[6] ca. $460\,\mathrm{km/h}$ oder $130\,\mathrm{m/s}$

Kollisionsgefahr zu bewerten. Um die Kollision zu vermeiden muss der Geschwindigkeitsvektor des Ownships so angepasst werden, dass der Vektor der Relativgeschwindigkeit nicht mehr auf den Intruder zeigt. Ein DAA Algorithmus unter Verwendung eines geometrischen Ansatzes wurde von Pascal Guillouet in einem dieser Arbeit vorangegangenen Praktikum beim DLR umgesetzt und bewertet.
Eine weitere Möglichkeit bietet die Betrachtung des Umfelds als ein Potentialfeld [54]. Das Ownship sowie mögliche Intruder werden durch gleich geladene, sich abstoßende, Teilchen repräsentiert. Das Ziel (zum Beispiel ein Wegpunkt) wird durch ein invers geladenes Teilchen repräsentiert. Durch die Anpassung der elektrostatischen Gleichungen kann so ein kontinuierliches Manöver vorbei an den Intrudern zum Ziel berechnet werden. [54]
Daneben existiert die Trajektorieoptimierung mit Hilfe des A*-Algorithmus [54, 65]. Beim A*-Algorithmus handelt es sich um einen Algorithmus, der den günstigsten (das bedeutet in der Regel den kürzesten) Pfad von einem Start- zu einem Zielpunkt findet. Dabei wird Schritt für Schritt jeder mögliche Pfad durch eine Kostenfunktion bewertet. Von Iteration zu Iteration werden die jeweils günstigsten Pfade betrachtet, bis der insgesamt günstigste Pfad gefunden wurde. [65]
Auch Neuronale Netze werden in der Erforschung möglicher Systeme zur Kollisionsvermeidung betrachtet. Ein solches System wurde bereits im Flugtest demonstriert. Das Neuronale Netz wird eingesetzt, um die Trajektorie des Ownships und des Intruders vorherzusagen und im Anschluss ein Ausweichmanöver zu planen. [17]
Die in dieser Arbeit zu betrachtende DAIDALUS Programmbibliothek nutzt für die Berechnung einer Ausweichempfehlung eine weitere Möglichkeit: Der Algorithmus von DAIDALUS betrachtet den Konfigurationsparametern entsprechend eine Vielzahl möglicher Flugrichtungen. Dabei wird die nächstgelegene Flugrichtung, von der die geringste Gefahr einer Kollision ausgeht, als Empfehlung ausgegeben [29]. Eine genauere Beschreibung der Funktionsweise der DAIDALUS Programmbibliothek ist in Unterkapitel 3.1 gegeben.

2.2.4. Bemannt

Im Folgenden werden verschiedene DAA Systeme vorgestellt, die bereits in der bemannten Luftfahrt eingesetzt werden oder aktuell für die bemannte Luftfahrt entwickelt werden.

TCAS

Für manntragende Luftfahrzeuge ist in Europa ab einer Masse von $5,7\,\mathrm{t}$ oder 19 Sitzplätzen das TCAS II vorgeschrieben [31, 18]. In den USA ist TCAS I Pflicht für Flugzeuge mit 10 bis 30 Sitzplätzen [1]. Ein Flugzeug mit mehr als 30 Sitzplätzen muss mit TCAS II ausgestattet sein [1].
Ein Luftfahrzeug muss mit einem funktionierenden Mode-S Transponder ausgerüstet sein, um TCAS nutzen zu können. Sind zwei mit TCAS ausgestattete Luftfahrzeuge auf Kollisionskurs, wird beiden Piloten eine RA - in diesem Fall ein Ausweichmanöver in entgegengesetzte Richtungen - befohlen. Bei TCAS handelt es sich somit um ein kooperatives System. Die *time of closest approach* (Zeit bis zum geringsten Abstand) und die *projected miss distance* (projizierte Distanz beim Vorbeifliegen) bilden die Basis für die Berechnung einer RA [34].

Die Berechnungen basieren auf den in den DO-185B spezifizierten *CAS logic functions* [35]. In einem Kollisionsszenario wird dem Piloten von Luftfahrzeug A beispielsweise vorgegeben zu steigen, während dem Piloten von Luftfahrzeug B vorgegeben wird zu sinken. TCAS greift jedoch nicht direkt in den Autopiloten der jeweiligen Luftfahrzeuge ein. Reagiert einer der beiden Piloten falsch, so wird der andere Pilot von seinem TCAS System angewiesen, in die entgegengesetzte Richtung auszuweichen. Dabei betrachtet TCAS nur ein Ausweichen in vertikaler Richtung - also durch Steigen beziehungsweise Sinken der jeweiligen Luftfahrzeuge. [35]

ACAS X

Die Entwicklung des ACAS X wurde durch die FAA gefördert. ACAS X stellt eine Familie an Software dar. Dabei stellt ACAS X_A den direkten (geplanten) Nachfolger von TCAS dar [45]. ACAS X_A ist wie TCAS ein kooperatives System und bietet ebenso lediglich eine RA in vertikaler Richtung. Eine Auswahl der weiteren (zum Teil in Zukunft erscheinenden) Varianten von ACAS X ist nachfolgend gegeben.

- ACAS X_P ist eine Version, die ADS-B Daten von anderen Luftfahrzeugen auswertet und ist für die Luftfahrzeuge der allgemeinen Luftfahrt gedacht, die aktuell nicht über TCAS II verfügen müssen. Das P steht für „passiv“. Mit der Verwendung eines passiven ADS-B Moduls kann ACAS X_P lediglich empfangene ADS-B Daten von anderen Luftfahrzeugen auswerten, selbst aber keine derartigen Daten ausgeben. Damit gehört ACAS X_P zu den nicht kooperativen Systemen.
- ACAS X_O ist ein Modus von ACAS X, der speziell für Szenarien angedacht ist, in denen ACAS X_A nicht sinnvoll eingesetzt werden kann. Beispielsweise beim dichten parallelen Landeanflug mehrerer Luftfahrzeuge. ACAS X_O gehört zur Gruppe der kooperativen Systeme.
- ACAS X_U ist für den Einsatz in unbemannten Luftfahrzeugen angedacht. ACAS X_U wird in Abschnitt 2.2.5 näher betrachtet.

ACAS X_A soll dieselbe Hardware wie TCAS II verwenden können. Eine Integration in bestehende Luftfahrzeug-Muster soll damit erleichtert werden. Auch die Anzeigen und Warnungen im Cockpit sind identisch mit denen von TCAS II. Eine große Umstellung auf das neue System wird der Crew damit erspart. ACAS X_A soll im Vergleich zu TCAS II in der Lage sein, eine geringere Höhendifferenz zwischen zwei Luftfahrzeugen zu akzeptieren, ohne eine Warnung auszugeben. Des Weiteren wird ACAS X_A nicht nur auf Daten der Transponder Mode-S zugreifen, sondern auch auf Satellitengestützte Navigation und ADS-B Funktionen für die Situationsbeurteilung zurückgreifen. Im Gegensatz zu TCAS II setzt ACAS X_A nicht auf eine Reihe von fest definierten Funktionen zum Berechnen einer RA. Stattdessen wird eine numerische Nachschlagetabelle genutzt, die mit Hilfe eines Wahrscheinlichkeitsmodells optimiert wurde. [34]

EGPWS

Während TCAS den Piloten eines Luftfahrzeugs vor einer drohenden Kollision mit einem anderen Luftfahrzeug warnt, wird der Pilot eines mit einem *Enhanced Ground Proximity Warning System (EGPWS)* ausgerüsteten Luftfahrzeugs vor einer zu schnellen oder zu dichten Annäherung an den Boden gewarnt. Im Falle einer Warnung wird ein Pull-Up Manöver befohlen. Die Grundlage für die Ermittlung der Gefahr einer zu schnellen oder zu dichten Annäherung an den Boden besteht aus Topologiedaten. Es werden fortlaufend die aktuell gemessenen Höhenwerte mit den in den Topologiedaten an der aktuellen Position gegebenen Daten verglichen. [40]

FLARM

Vor allem im Bereich der leichten Luftfahrzeuge - sowohl bemannt, als auch unbemannt - hat sich in den vergangenen Jahren FLARM als Standard zur Kollisionsvermeidung etabliert. Mit FLARM ausgerüstete Luftfahrzeuge tauschen die eigenen, über ein GNSS Modul erhaltenen Positions- und Geschwindigkeitsdaten, mit anderen FLARM-fähigen Luftfahrzeugen aus. Steht eine Kollision zwischen zwei Luftfahrzeugen bevor, so werden die Piloten der jeweiligen Luftfahrzeuge unter Angabe der relativen Position des anderen Luftfahrzeugs gewarnt. FLARM teilt lediglich die Positions- und Geschwindigkeitsdaten der jeweiligen Luftfahrzeuge. Eine Ausweichempfehlung wird von FLARM nicht bereitgestellt. Damit bietet FLARM im Bereich der Erkennung von anderen Luftfahrzeugen zwar kooperatives Verhalten, im Bereich der Vermeidung handelt es sich aufgrund der fehlenden Ausweichempfehlung allerdings um ein nicht kooperatives Verhalten. Die Piloten müssen selbst aktiv die Flugrichtung ändern um die Kollision zu verhindern. Dabei sollten sie die in Abschnitt 2.1.1 vorgestellten Regeln für das Ausweichmanöver anwenden und die Reaktion des anderen Luftfahrzeugs beobachten. [7]

2.2.5. Unbemannt

Im Folgenden werden verschiedene Systeme zur Kollisionsvermeidung vorgestellt, die Anwendung in unbemannten Luftfahrzeugen finden oder zum Zeitpunkt der Veröffentlichung dieser Arbeit dafür entwickelt werden.

DAIDALUS

DAIDALUS wurde als Referenzimplementierung der DO-365A [43] erstellt. DAIDALUS verarbeitet die vom Luftfahrzeug bereitgestellten Positions- und Geschwindigkeitsdaten des Ownships und aller bekannten fremden Luftfahrzeuge. Basierend auf dieser Momentaufnahme berechnet DAIDALUS ob eine Kollisionsgefahr innerhalb der gewählten Parameter besteht. Im Falle einer Kollisionsgefahr berechnet DAIDALUS die nächstgelegene Flugrichtung, um den fremden Luftfahrzeugen auszuweichen. Dabei kann DAIDALUS eine Ausweichrichtung sowohl durch eine Änderung des Kurses, der Vertikal- sowie der Horizontalgeschwindigkeit berechnen. Da das berechnete Ausweichmanöver nicht mit dem Intruder geteilt wird, handelt es sich bei DAIDALUS um ein nicht kooperatives System. Nach Ausgabe der von DAIDALUS empfohlenen Ausweichrichtung kann diese vom Fernpiloten beziehungsweise vom Autopiloten umgesetzt werden. [61]

Auch so genannte Bänder können von DAIDALUS ausgegeben werden. Die Bänder beschreiben Bereiche in Kurs, horizontaler Geschwindigkeit und vertikaler Geschwindigkeit. Liegt der jeweilige Wert des Ownships innerhalb dieser Bänder, so droht eine Kollision mit dem bekannten Intruder. Die Bänder können dem Fernpiloten an einer GCS angezeigt werden und so die Beurteilung über mögliche Kollisionen vereinfachen. [61]
DAIDALUS ist seit Version 2 in der Lage, Sensorunsicherheiten und Winddaten in die Berechnung der Ausweichrichtung aufzunehmen [57, 61]. Allerdings wird in einer Veröffentlichung von Jason T. Davies et. al [27] von Ausfällen der Warnungen bei zu starkem Rauschen der Sensoren berichtet. Sensorunsicherheiten werden in dieser Arbeit nicht betrachtet.

ACAS X_U

Bei ACAS X_U handelt es sich um ein kooperatives System, sofern ein Transponder Mode-S vorhanden ist. Verglichen mit ACAS $X_{(A)}$ verfügt ACAS X_U über die Fähigkeit, auch in horizontaler Richtung ausweichen zu können. Dabei werden die Berechnungen zum horizontalen und vertikalen Ausweichen unabhängig voneinander betrachtet. Die Berechnungen werden durch Lösen eines Markov-Entscheidungsproblems getätigt. [62]
Ein Vergleich zwischen ACAS X_U und DAIDALUS (in der Version v1.0.1) zeigte, dass ACAS X_U früher vor möglichen Kollisionen warnt [27]. Außerdem verarbeitet ACAS X_U Sensorunsicherheiten zuverlässiger als DAIDALUS [27]. Da die Verarbeitung von Sensorunsicherheiten erst in der zweiten Version von DAIDALUS implementiert wurde [61], ist dieses Ergebnis für aktuelle Vergleiche beider Systeme nicht belastbar.

ICAROUS

ICAROUS (Independent Configurable Architecture for Reliable Operations of Unmanned Systems) ist ein Forschungsprojekt der NASA. Es handelt sich um eine modular aufgebaute Softwarearchitektur, die mehrere Softwaremodule vereint. Die Module werden parallel als Programmanwendung des von der NASA entwickelten core Flight Systems (*cFS*) ausgeführt. Durch das modulare Konzept ist die Software sehr flexibel anpassbar. Die Module bieten beispielsweise Funktionen zur Flugpfadplanung, zur Einhaltung von Geofences oder auch eine Schnittstelle zu einer Bodenstation. Für die Realisierung einer DAA Funktion wird DAIDALUS als Anwendung auf dem cFS ausgeführt. Das Einhalten von Geofences wird durch die Verwendung von PolyCARP (Algorithms and Software for Computations with Polygons) [60] gewährleistet. ICAROUS selbst bietet keine Möglichkeit zur direkten Steuerung eines UAs. Allerdings wird eine Schnittstelle zur Verfügung gestellt, die die Kommunikation mit verschiedenen etablierten Autopiloten ermöglicht.
Die einzelnen Module werden unabhängig voneinander ausgeführt und kommunizieren über einen Publish/Subscribe Service. Es wurde gezeigt, dass ICAROUS auch auf Plattformen mit geringer Rechenleistung eingesetzt werden kann [59, 11]

PX4 Autopilot

PX4 ist ein Open Source Autopilot für verschiedene Konfigurationen von UA aber auch für unbemannte Bodenfahrzeuge sowie unbemannte U-Boote [25]. PX4 kann unter anderem FLARM und ADS-B Daten verarbeiten und darauf basierend bei drohenden Warnungen au-

tomatisiert landen, zu einem zuvor definierten Wegpunkt fliegen oder lediglich eine Warnung an den Piloten/die Bodenstation abgeben. Ein kooperatives Verhalten ist nicht gegeben. Besteht die Kollisionswarnung nicht mehr, so verharrt das UA an der jeweiligen Position. [24]
Vor dem Flug ist es möglich, Geofences für PX4 vorzugeben, in die entweder nicht eingeflogen oder aus denen nicht herausgeflogen werden darf. Die Geofences können aus Polygonen oder Kreisen bestehen. [26]
Außerdem lässt sich PX4 mit einem nach unten ausgerichteten Distanzsensor verbinden. Ein damit ausgestattetes UA kann dem Geländeverlauf folgen und einen konstanten Abstand zum Boden einhalten. [53]

Ardupilot

Ardupilot ist wie PX4 ein Open Source Autopilot für verschiedene Konfigurationen von UA aber auch für unbemannte Bodenfahrzeuge sowie unbemannte U-Boote [22]. Ardupilot kann ADS-B Daten nutzen um sich annähernden Luftfahrzeugen auszuweichen. Dies kann wahlweise in horizontaler, vertikaler oder kombinierter, dreidimensionaler Richtung geschehen. Das Ausweichmanöver wird dabei mit einem dem A*-Algorithmus ähnlichen Vogehen berechnet. Alternativ kann dem UA (im Falle einer Multicopter-Konfiguration) auch ein Schwebeflug befohlen werden. Außerdem kann gewählt werden, ob der geplante Flug nach dem Ausweichmanöver fortgesetzt oder abgebrochen werden soll. Wie bei PX4 ist auch bei Ardupilot kein kooperatives Verhalten gegeben. [23]
Ardupilot kann verschiedene Arten von Geofences verarbeiten. Bei Ardupilot besteht ein Geofence entweder aus Kreisen oder Polygonen. Die Geofences können entweder als Inclusion oder Exclusion Zone definiert werden, erlauben allerdings keine Höhenbeschränkung/-Freigabe pro Geofence. [20]
Auch Ardupilot verfügt über die Möglichkeit, dem Geländeverlauf zu folgen. Der Terrain Follow Modus orientiert sich bei Ardupilot wahlweise an einem nach unten gerichteten Distanzsensor, oder greift auf eine an der Bodenstation vorhandene Datenbank mit topologischen Daten zurück. [21]

Paparazzi UAV

Ein weiterer Open Source Autopilot für UA, der Flächen- und Rotorkonfigurationen unterstützt, ist Paparazzi UAV. Paparazzi UAV bietet ein an TCAS angelehntes Ausweichsystem. Die Daten der anderen mit Paparazzi UAV ausgestatteten UA erhält das Ownship über einen Datenlink zur GCS. Wird ein anderes UA auf Kollisionskurs registriert, so weicht das höher fliegende UA nach oben und das tiefer fliegende UA nach unten aus. Die vom Ownship berechnete RA wird über die GCS an das andere UA gesendet. Damit handelt es sich um ein kooperatives System. Während das Ausweichmanöver geflogen wird gleicht der Autopilot fortlaufend die Höhe des anderen UAs ab. Erkennt das Ownship die gleiche Ausweichrichtung wie bei sich selbst (z. B. beide UA steigen), so wird die eigene Ausweichrichtung korrigiert.
Paparazzi UAV ermöglicht die Verwendung von verschiedenen Sensoren. Ein Infrarot Sensor ermöglicht die Messung der Flughöhe über dem Gelände. Ein Luftdrucksensor macht eine

Höhenberechnung anhand der Änderung des Luftdrucks möglich. Darüber hinaus werden GPS-, Gyroskop-, IMU- und Staudruckdensoren unterstützt. [63]

MIDCAS/EUDAAS

Im Rahmen des MIDCAS (Mid-Air Collission Avoidance System) Projekts wurde in einem Konsortium aus fünf europäischen Staaten (Frankreich, Deutschland, Italien und Spanien unter der Führung von Schweden) ein DAA System entwickelt, das in einem ferngesteuerten Luftfahrsystem (engl. *Remotely Piloted Aircraft System* (RPAS)) ab Dezember 2014 getestet wurde [8]. Das MIDCAS nutzte dabei sowohl kooperative (ADS-B) als auch nicht-kooperative (elektro-optische, Infrarot- und Radarsensoren) Erkennungssysteme und steuerte das RPAS vollautomatisch ohne nötigen Eingriff des Fernpiloten [8]. Das EUDAAS (European Detect and Avoid System) Projekt ist das Nachfolgeprojekt von MIDCAS. Es soll die Integration von großen militärischen RPAS in den Luftraum weiter vorantreiben. [9]

2.2.6. Unterschiede und Gemeinsamkeiten

Sowohl die DAA Systeme der bemannten als auch der unbemannten Luftfahrzeuge können Warnungen zu bevorstehenden Kollisionen ausgeben. Dabei wird im Falle von TCAS II (und später ACAS $X_{(A)}$) im Cockpit nur ein vertikales Ausweichmanöver angeordnet [35, 34]. Dagegen geben die DAA Systeme der UA in der Regel ein Ausweichmanöver über eine Änderung der Horizontal- und Vertikalgeschwindigkeit sowie des Kurses an den Autopiloten weiter [58, 27, 23].

Ein weiterer Unterschied besteht in der Kommunikation zwischen Ownship und Intruder. Aktuelle DAA Systeme für UA kommunizieren nicht untereinander sondern berechnen das Ausweichmanöver unter der Annahme eines konstanten Geschwindigkeitvektors des Intruders [58, 23]. Bei TCAS II (und später ACAS $X_{(A)}$) wird bei Erkennen einer möglichen Kollision eine Master/Slave-Kommunikation zwischen beiden Luftfahrzeugen unterhalten [35, 34]. Wird im Cockpit eines der Luftfahrzeuge nicht oder falsch reagiert, so wird dies dem anderen Luftfahrzeug mitgeteilt und die entgegengesetzte Ausweichrichtung angewiesen [35].

2.3. (Quasi-)Statische Hindernisse und Geofences

Neben anderen Luftfahrzeugen muss für Überlandflüge, wie sie mit dem ALAADy-Demonstrator geplant sind, auch die Beachtung von anderen Hindernissen betrachtet werden. Die im Folgenden exemplarisch aufgezeigten Hindernisse sind entweder statisch oder – im Vergleich zu Luftfahrzeugen – so langsam, dass sie im Kontext dieser Arbeit als quasi-statische Objekte betrachtet werden können. Viele der (quasi-)statischen Hindernisse können bereits vor dem eigentlichen Flug des unbemannten Luftfahrzeugs in der Missionsplanung betrachtet werden. Eine Reihe von Datenbanken, aus denen die nötigen Informationen gewonnen werden können, werden in den folgenden Abschnitten vorgestellt. Darüberhinaus ist zu beachten, dass es auch mobile statische Hindernisse (wie zum Beispiel Kräne) gibt, deren Position und Dimensionen nicht immer vor dem Flug bekannt sind. Im Rahmen dieser Arbeit werden

generische Hindernisse innerhalb einer Simulationsumgebung behandelt. Die generischen Hindernisse sind nicht Bestandteil einer der nachfolgend vorgestellten Datenbanken.

2.3.1. Gelände

Im Rahmen der Shuttle Radar Topography Mission (SRTM) wurde im Februar 2000 ein digitales Höhenmodell der Erde erstellt. Durchgeführt wurde das Projekt von der NASA in Kooperation mit der National Imagery and Mapping Agency (NIMA) [15]. Im August 2015 wurden die Daten in der hohen Auflösung von einer Bogensekunde (etwa 30m) von der NASA veröffentlicht [15]. Mittlerweile werden die Daten zum Beispiel auch zur Erstellung der OpenTopoMap genutzt und können von dort in verschiedenen Formaten heruntergeladen werden [32].
Das DLR hat die im Rahmen der TanDEM-X-Mission gewonnenen Höhendaten mittlerweile zum kostenlosen Download bereitgestellt. Die Daten stammen aus den Jahren 2010 bis 2015 und werden fortlaufend gepflegt. Sie bieten eine horizontale Auflösung von 90 Metern und eine Höhengenauigkeit von bis zu einem Meter. [49]
Daneben bestehen noch zahlreiche andere Quellen für Höhendaten. Je nach verwendeter Auflösung und Messmethode unterscheiden sich diese in ihrem Detailgrad. Eine vollständige Aufzählung der einzelnen Quellen würde im Rahmen dieser Arbeit zu weit führen.
Wie in Abschnitt 2.2.5 aufgezeigt, sind lediglich PX4 und Ardupilot in der Lage, dem Geländeverlauf zu folgen. Die anderen in Abschnitt 2.2.5 vorgestellten Systeme bieten diese Möglichkeit nicht.

2.3.2. Wetter

Aktuelle Wettermessdaten, Warnungen und weitere interessante Informationen bietet das Open-Data-Angebot des Deutschen Wetterdienstes (DWD) [68]. Über das Open-Data-Angebot werden sowohl aktuelle Messwerte von Wetterstationen in Deutschland geteilt als auch Warnungen zu drohenden Unwettern ausgegeben. Für die Nutzung eines unbemannten Luftfahrzeugs von besonderem Interesse sind die aktuellen Messwerte der Windgeschwindigkeiten sowie die Daten zu Mesozyklonen aus Radarüberwachungen. Von vorhandenen Mesozyklonen lässt sich auf die Präsenz von Gewitterzellen schließen [69]. Eine übersichtliche Suche für die einzelnen Module bietet das DWD-Geoportal [67]. In aktuellen Forschungsprojekten werden vor allem elektro-optische und radarbasierte Sensoren genutzt um auf die Präsenz von Wolken zu schließen [28]. Nach aktuellem Stand bietet kein in 2.2.5 vorgestelltes System die Möglichkeit, bestimmte Wetterereignisse reaktiv zu meiden.

2.3.3. Lufträume

Wie alle anderen Luftfahrzeuge müssen auch automatisiert fliegende UA bestimmte Lufträume umfliegen. Lufträume sind generell in ihrer vertikalen Ausprägung beschränkt. In einer Höhe von maximal $500\,\mathrm{ft\,AGL}$ befindet sich in Deutschland in der Regel der unkontrollierte Luftraum G. Um Flughäfen befindet sich in der Regel ein imaginärer Trichter aus anderen Lufträumen – in Deutschland Luftraum C/D. Die Lufträume C und D dürfen nach aktuellem Stand in der Regel nicht von UA genutzt werden und müssen umflogen werden. Darüber hinaus müssen auch andere Gebiete, wie zum Beispiel militärische Sperrgebiete (ED-R) ge-

mieden werden. [36]
Die Daten zur aktuell vorherrschenden Luftraumstruktur sind auf unterschiedliche Art zugänglich. Die Quelldaten stammen dabei von der DFS [44]. Für die Nutzung eines UAs sind die elektronischen Datensätze praktisch. Die Datensätze können beispielsweise im OpenAir Format auf der Internetseite des Deutscher Aero Club e.V. (DAEC) kostenlos heruntergeladen werden [30]. Die Daten werden im Zyklus von sechs Monaten aktualisiert. Daten zu kurzfristig aktivierten oder geänderten Lufträumen sind kostenlos über das AIS-Portal der DFS erhältlich [36]. Hier ist vor allem der Abschnitt ENR 2 interessant. ENR 2 bietet Informationen, die bei der Planung von Streckenflügen zu beachten sind und beinhaltet somit auch die kurzfristig aktivierten und geänderten Lufträume. [36]
Die so erhaltenen Daten zu den jeweiligen Daten können entsprechend gefiltert und in Geofences umgewandelt werden. Die Geofences können dann beispielsweise von Flugplanungsprogrammen (engl. *Flight Management System* (FMS)) und Autopiloten wie PX4 und Ardupilot beachtet werden.

2.3.4. Stromleitungen, (Material-)Seilbahnen

Stromleitungen und (Material-)Seilbahnen stellen bodennahe Objekte dar und müssen daher bei der Betrachtung von Hindernissen für tieffliegende UA beachtet werden. Daten sind sowohl über OpenStreetMap (OSM) [38] (kostenlos) als auch über die FLARM Hindernisdatenbank [5] (kostenpflichtig) zu erhalten. Einige Veröffentlichungen behandeln die Erkennung von Stromleitungen auf der Basis von elektro-optischen Sensoren [46, 47]. Durch die Aufbereitung der Bilddaten werden die Stromleitungen von anderen parallelen Linien (zum Beispiel Straßenrändern [46]) unterschieden und erkannt. Die Erkennung wird in den Veröffentlichungen [46, 47] jedoch nicht zum Umfliegen der Hindernisse genutzt, sondern mit dem Hintergrund, die Stromleitungen kostengünstig zu inspizieren.

2.3.5. Bauwerke

Wie alle (quasi-)statische Hindernisse sind auch Gebäude, Türme und andere Bauwerke nicht mit einem FLARM/ADS-B Sender ausgestattet und können von einem UA ohne elektro-optische Sensoren, Radarsensoren oder anderen Sensoren daher nicht erkannt werden. Auch in topographischen Karten sind diese in der Regel nicht hinterlegt.
Eine Datenbank zu Gebäuden lässt sich beispielsweise mit den Daten von OSM [38] generieren. Zu bedenken ist hierbei allerdings, dass es sich bei OSM um ein OpenSource Projekt handelt. Durch die unterschiedliche Aktivität der OSM Community in den jeweiligen Regionen, kann die Abdeckung und Vollständigkeit der Daten regional sehr unterschiedlich sein [39]. Zu beachten ist auch, dass die Daten der Gebäude lediglich den 2D-Grundriss darstellen. Die Höhe der Gebäude ist im Allgemeinen unbekannt. [38]
Die Deutsche Flugsicherung bietet über das AIS-Portal unter „AIXM Datensätze“ eine umfangreiche Datenbank mit Türmen, Windrädern und ähnlichen Bauwerken an [36]. Die Vollständigkeit dieser Daten kann nicht überprüft werden.
Bei einer Flughöhe bis $500\,\mathrm{ft\,AGL}$ gilt es auch temporäre Objekte wie zum Beispiel Baukräne zu beachten. Hierzu sind keine Datenbanken bekannt.

3. Methode

In diesem Kapitel wird zunächst eine Einführung in die Funktionsweise und die erstellte Anwendung der Programmbibliothek DAIDALUS gegeben. Anschließend wird in Unterkapitel 3.2 der Aufbau des ALAADy-Missionsmanagers vorgestellt. Mit der DAIDALUS-ALAADy (DAL)-Bridge wird die im Rahmen dieser Arbeit entwickelte Kommunikationsschnittstelle zwischen dem ALAADy-Demonstrator und DAIDALUS präsentiert. Anschließend werden die im Rahmen dieser Arbeit getroffenen Annahmen genannt. Das Kapitel schließt mit der Beschreibung der in dieser Arbeit verwendeten Simulationsumgebungen in Unterkapitel 3.3.

3.1. Programmbibliothek DAIDALUS

In diesem Unterkapitel wird die Funktionsweise der DAIDALUS Programmbibliothek genauer betrachtet. Anschließend wird ein Pseudocode für die im Rahmen dieser Arbeit verwendete Implementierung gegeben.

3.1.1. Funktionsweise

Eine ausführliche - wenn auch zum Zeitpunkt der Veröffentlichung dieser Arbeit unvollständige - Dokumentation von DAIDALUS ist im Internet verfügbar [61]. Dieser Abschnitt gibt eine kurze Übersicht über die Funktionsweise von DAIDALUS.
DAIDALUS ist eine Referenz Implementierung der von der RTCA veröffentlichten *Minimum Operational Performance Standards* DO-365 und DO-365A [42, 43]. Wie in Abschnitt 2.2.1 beschrieben werden mehrere zylindrische Volumen um die gegebenen Positionen jedes einzelnen fremden Luftfahrzeugs oder des Ownships erzeugt. Die Dimensionen, die jedes zylindrische Volumen einnimmt, kann durch die sogenannten Alerter definiert werden. Die hierfür genutzten Bezeichnungen der Werte wurde in Tabelle 2.1 vorgestellt. Im Rahmen dieser Arbeit wird (sofern nicht anders beschrieben) der `DWC_Phase_I_SUM`-Alerter genutzt. Er ist in der `DO_365A_SUM.conf` Konfigurationsdatei enthalten. Die Konfigurationsdatei kann über das GitHub Repository von DAIDALUS [58] heruntergeladen werden. Die Alerter können - ebenso wie zahlreiche andere Parameter - entweder im Programmcode mittels fest definierten Funktionsaufrufen bearbeitet werden, oder durch das Laden einer Konfigurationsdatei definiert werden. Das Laden einer Konfigurationsdatei erlaubt eine schnelle und übersichtliche Arbeitsweise. Einzelne Parameter können so komfortabel mit dem Ändern einer Textdatei vor der Ausführung des Programmcodes angepasst werden. Damit bleibt der Programmcode übersichtlich und muss nicht vor jedem einzelnen Simulationsdurchlauf neu kompiliert werden. Die im Rahmen dieser Arbeit verwendete Konfigurationsdatei ist im Anhang A zu finden.
Ist die Option `ownship_centric_alerting` in der geladenen Konfigurationsdatei auf *true* gesetzt, wird das durch den Alerter definierte zylindrische Volumen um das Ownship aufgespannt und gilt damit für jedes fremde Luftfahrzeug. Ist die Option *ownship_centric_alerting* in der geladenen Konfigurationsdatei auf *false* gesetzt, so kann jedem fremden Luftfahrzeug

ein eigener Alerter zugeordnet werden. In dieser Arbeit ist der Wert *ownship_centric_alerting* zu jeder Zeit auf *false* gesetzt.
Zur Überprüfung, ob eines der zylindrischen Volumen innerhalb der definierten *lookahead_time* T verletzt wird, nutzt DAIDALUS intern ein East-North-UP (ENU) Koordinatensystem. Die Lookahead Time bezeichnet den Zeithorizont, den DAIDALUS in den Berechnungen berücksichtigt. DAIDALUS projiziert die Positionen des Ownships und der fremden Luftfahrzeuge auf eine parallel zur lokalen Erdoberfläche gelegenen Fläche. Die Höhe wird dabei nicht transformiert [61]. DAIDALUS nutzt intern zwar ein ENU Koordinatensystem, das Bogenmaß und die Maßeinheiten Meter und Sekunden. Positions- und Geschwindigkeitsdaten sollen jedoch bevorzugt im WGS84 System und mit Kurs (engl. Track), Fluggeschwindigkeit über Grund (engl. Groundspeed) und Vertikalgeschwindigkeit (engl. Verticalspeed) übergeben werden [61].
Je nachdem, welches der Volumen innerhalb der Lookahead Time verletzt wird, sind unterschiedliche Reaktionen von DAIDALUS möglich. Dazu sind in DAIDALUS verschiedene Regionen definiert (*NONE, FAR, MID, NEAR, RECOVERY*) [61]. Standardmäßig wird durch Verletzung eines Volumens der Region *MID* ein so genannter *Corrective Alert* ausgegeben. Bei Auftreten eines *Corrective Alerts* wird eine Reaktion des (Fern-)Piloten gefordert [43]. Die Vorgabe durch den Standard wird im Rahmen dieser Arbeit als Trigger genutzt, ein Ausweichmanöver zu berechnen. Zur Berechnung einer Ausweichführung (Maneuver Guidance[1]) prüft DAIDALUS basierend auf den aktuellen Positions- und Geschwindigkeitsdaten mögliche Ausweichrichtungen. Dabei wird jede der vier möglichen Dimensionen (Kurs, horizontale Geschwindigkeit, vertikale Geschwindigkeit, Höhe) unabhängig voneinander betrachtet. Überprüft wird, mit welchen Werten auf der jeweiligen Dimension die geringste Zeit in einem zu vermeidenden Volumen verbracht wird. Zur Beurteilung der Flugbahn wird dazu ein kinematisches Bewegungsmodell genutzt. Der nachfolgende Pseudocode stellt die Berechnung eines Ausweichmanövers in horizontaler Richtung nach links vor. Der Pseudocode und die entsprechenden Codebeispiele für die anderen Ausweichdimensionen sind in einer Veröffentlichung zu DAIDALUS zu finden [56].

```
1  left_1x1(s,v,s_z,v_z,B,T,c,u_min,u_max,e,a,D,H,p̄,v̄):set[ℝ²]≡
2      let t_e = e/a in
3      t := 0;
4      u := c;
5      β := ∅;
6      while u<u_max and t<T do
7          (s_t, s_zt)=p̄(s,v,a,t);
8          (v_t, v_zt)=v̄(s,v,a,t);
9          if t<B then
10             if ‖s_t‖ < D and |s_zt|<H then
11             β := β ∪ {[u,u_max]};
12             u := u_max;
```

[1] in anderen DAA Systemen auch als RA bezeichnet

```
13              endif
14          elseif WCV(s_t ,v_t ,s_zt ,v_zt) then
15              β := β ∪ {[u,u_max]};
16              u := u_max ;
17          endif
18          elseif WCV(s_t ,v_t ,s_zt ,v_zt,0,T-t) then
19              β := β ∪ {[u,u+e]};
20              u := u+e;
21          endif
22          t:=t+t_e ;
23      endwhile
24      return β;
```

Die Variablen sind wie folgt definiert:

- **s**: Differenzvektor der Positionsvektoren des Ownships und des Intruders in der horizontalen Ebene (**s** = $\mathbf{s}_o$ - $\mathbf{s}_i$)
- **v**: Differenzvektor der Geschwindigkeitsvektoren des Ownships und des Intruders in der horizontalen Ebene (**v** = $\mathbf{v}_o$ - $\mathbf{v}_i$)
- s_z: Höhendifferenz zwischen Ownship und Intruder ($\mathbf{s}_z$ = $\mathbf{s}_{zo}$ - $\mathbf{s}_{zi}$)
- v_z: Vertikalgeschwindigkeit zwischen Ownship und Intruder ($\mathbf{v}_z$ = $\mathbf{v}_{zo}$ - $\mathbf{v}_{zi}$)
- $[B, T]$: Zeitintervall der Lookahead Time T , wobei B in der Regel mit 0 definiert ist
- c: aktueller Manöverwert des Ownships (in obigem Beispiel der Kurs des Ownships)
- u_{min},u_{max}: Minimal-/Maximalwert des Manövers (in obigem Beispiel kann damit beispielsweise ein Ausweimanöver mit einer Änderung des Kurses von bis zu $20°$ erlaubt werden)
- e: Manöverschritt
- a: Beschleunigung des Manövers
- D,H: horizontale und vertikale Distanzschwellwerte
- $\bar{p}$,$\bar{v}$: Positions- und Geschwindigkeitsfunktionen, die das kinematische Modell mit konstanter Beschleunigung beinhalten

In Zeile 6 des Pseudocodes wird für jeden Zeitpunkt t innerhalb der Lookahead Time T und solange die erlaube Manövergrenze u_{max} nicht überschritten ist, nach einer Lösung gesucht. Dazu werden die zum Zeitpunkt t zu erwartenden Differenzvektoren der Positions- und Geschwindigkeitsvektoren ($\mathbf{s}_t$, $\mathbf{v}_t$) mit Hilfe des kinematischen Bewegungsmodells berechnet. Sofern t kleiner als B ist - und damit nicht im Intervall der zu betrachtenden Lookahead Time

- liegt, wird die zu Beginn leere Menge an Lösungen β in Zeile 11 um die Menge {[u,u_{max}]} erweitert und u umgehend auf u_{max} gesetzt. Damit wird die While-Schleife nicht weiter ausgeführt. Liegt der Zeitschritt t dagegen im Intervall [B,T] wird geprüft, ob mit den berechneten Positions- und Geschwindigkeitswerten zum Zeitpunkt t eine WCV besteht. Besteht mit den in Zeile 7 und 8 berechneten Positions- und Geschwindigkeitswerten eine WCV, so werden die gleichen Aktionen wie in Zeile 11 und 12 ausgeführt. Ist (vom Zeitpunkt t aus betrachtet) innerhalb der Lookahead Time eine WCV zu erwarten (Zeile 18 ist wahr), so wird die Lösungsmenge β um die Menge {[u,u+e]} erweitert und u um einen Manöverschritt erhöht (Zeilen 19 und 20). Am Ende eines Durchlaufs der While-Schleife wird der Zeitpunkt t um einen Zeitschritt t_e erhöht. Wenn die While-Schleife beendet wird, wird die gesammelte Lösungsmenge β zurückgegeben.
Bei der Implementierung von DAIDALUS ist darauf zu achten, dass die in der Konfigurationsdatei geladenen Variablen *recovery_hdir*, *recovery_hs*, *recovery_vs* und *recovery_alt* nur beachtet werden, wenn eine Berechnung einer Recovery Guidance gefordert ist. Das heißt, wenn eine Verletzung des *Corrective Volumes* durch eine Maneuver Guidance nicht mehr gewährleistet werden kann. Soll beispielsweise nur (wie in dieser Arbeit) durch eine horizontale Richtungsänderung ausgewichen werden, darf lediglich die Ausweichempfehlung von DAIDALUS in dieser Dimension aktiv übernommen werden. Alle anderen Werte (im Rahmen dieser Arbeit Horizontalgeschwindigkeit, Vertikalgeschwindigkeit und Höhe) müssen auf andere Weise vorgegeben werden.
Außerdem kann es dazu kommen, dass die von der Konfigurationsdatei geladenen Flugleistungsgrenzen des Ownships nicht eingehalten werden. Aaron M. Dutle - einer der Entwickler von DAIDALUS - bestätigte, dass es bei der ersten Berechnung der Guidance zu Sprüngen kommen kann, die die Flugleistungsgrenzen des Ownships (also die Minimal-/Maximalwerte für die Fluggeschwindigkeit, Vertikalgeschwindigkeit und Wenderate) übersteigen. Abgesehen von einem ersten Sprung bei der ersten Berechnung haben die Entwickler die Einhaltung der Flugleistungsgrenzen in diversen Simulationen getestet und bestätigt. [29]

3.1.2. Anwendung

Der folgende Pseudocode zeigt die im Rahmen dieser Arbeit umgesetzte Implementierung von DAIDALUS. Kommentare sind mit // gekennzeichnet. Der Präfix *o_* wird für Variablen des Ownships genutzt. Der Präfix *h_* wird für Variablen der (unterschiedlichen) Hindernisse genutzt.

```
1 DAIDALUS::loadFromFile(path_to_config_file)
2 //if hardcoded velocities/directions are wanted, define them here
3
4 //ownship can be initialized with dummy data here
5 //however correct o_id is important!
6 DAIDALUS::setOwnship(o_id, o_pos, o_vel)
7
```

```
8  //to make sure that (current_time-alert_since)>10seconds is true
9  //on the first cycle, when there is no traffic yet
10 set_alert_since(0)
11
12 WHILE TRUE
13     IF UPDATERATE
14         get_Ownship_Information()
15         DAIDALUS::setOwnshipState(o_id,o_pos,o_vel,current_time)
16
17         FOR Traffic
18             get_Traffic_Information()
19             DAIDALUS::addTrafficState(h_id,h_pos,h_vel,current_time)
20             DAIDALUS::set_alerter_index(h_id,h_alerter)
21         ENDFOR
22
23         FOR Static Obstacles
24             get_Obstacle_Information()
25             DAIDALUS::addTrafficState(h_id,h_pos,h_vel,current_time)
26             DAIDALUS::set_alerter_index(h_id,h_alerter)
27         ENDFOR
28
29         /*
30         * alertLevel (as defined in DO-365A):
31         * 1: Preventive
32         * 2: Corrective
33         * 3: Warning
34         */
35         IF DAIDALUS::alertLevelAllTraffic()>1
36             calculate_guidance() //for all bands
37             check_for_performance_limits() //limit guidance if needed
38             save_guidance_to_array(guidance[])
39             //set not allowed bands in above array
40             //to current readings
41             set_alert_since(current_time)
42             DAA_FLAG = TRUE
43         ELSE
44             // t_sec is defined as the minimum duration the ownship
45             //should just fly ahead with the latest guidance
46             IF (current_time - alert_since) > t_sec
47                 DAA_FLAG = FALSE
48                 set_guidance_in_array_to_original_values(guidance[])
```

```
49                  //as DAA State will be left after this iteration
50                  //providing a guidance is optional
51              ELSE
52                  get_previous_calculated_guidance()
53                  save_guidance_to_array(guidance[])
54                  //there is no alert, but as the alert was active
55                  //within the last t_sec seconds, DAA_FLAG stays true
56                  DAA_FLAG = TRUE
57              ENDIF
58          ENDIF
59          set_DAA_FLAG(DAA_FLAG)
60      ENDIF
61      command_guidance_from_array(guidance[])
62  ENDWHILE
```

Wie der obige Code zeigt wird zu Beginn die Konfigurationsdatei (`path_to_config_file`) geladen und anschließend eine Ownship-Instanz mit Pseudodaten definiert. Die in Zeile 12 beginnende While-Schleife wird von hieran kontinuierlich durchlaufen. Sofern die Bedingung der Aktualisierungsrate von $f = 1\,\mathrm{Hz}$ wahr ist, werden die aktuellen Daten des Ownships an das DAIDALUS Objekt übergeben, ehe die bekannten Daten zu anderen Luftfahrzeugen und statischen Hindernissen übergeben werden.
Sofern eine Warnung von DAIDALUS erkannt wird (Zeile 35 ist wahr), wird eine Maneuver Guidance berechnet, die Einhaltung der Flugleistungsgrenzen des ALAADy-Demonstrators überprüft und die so erhaltene Guidance in einem Array `guidance[]` abgespeichert. Überschreitet die von DAIDALUS berechnete Guidance die Flugleistungsgrenzen des ALAADy-Demonstrators, so wird diese entsprechend den Flugleistungsgrenzen innerhalb der Aktualisierungsrate limitiert. Außerdem wird die Bool-Variable `DAA_FLAG` auf wahr gesetzt. Wird keine Warnung von DAIDALUS ausgegeben (Zeile 35 ist unwahr), so wird in Zeile 45 geprüft, ob die letzte Warnung seit über $\mathtt{t}_{sec}$ nicht mehr aktiv ist. Wird die Warnung seit über $\mathtt{t}_{sec}$ nicht ausgegeben (Zeile 45 ist wahr), wird die Bool-Variable `DAA_FLAG` auf unwahr gesetzt und die zu Beginn des Fluges geflogene Richtung in das Array `guidance[]` übernommen. Sofern die Zeit $\mathtt{t}_{sec}$ noch nicht abgelaufen ist (Zeile 45 ist unwahr), wird die im vorherigen Zeitschritt berechnete Maneuver Guidance in das Array `guidance[]` übertragen und die Bool-Variable `DAA_FLAG` auf wahr gesetzt. Wären die Codezeilen 45 bis 49 nicht gegeben, würde das Ownship beim Anflug auf ein Hindernis (und einer Warnung von DAIDALUS) wie gewünscht ausweichen. Sobald die Warnung nicht mehr existent wäre, würde dann wieder auf die ursprüngliche Richtung (`DAA_FLAG=flase`) zurückgedreht werden, was eine erneute Warnung von DAIDALUS zur Folge hätte, da das Ownship wieder auf Kollisionskurs mit dem Hindernis käme. Dieses Verhalten würde sich fortlaufend wiederholen und damit zu einem Zick-Zack-Flug entlang des *Corrective Volumes* führen. Durch diesen Zick-Zack-Flug kann

es im realen Betrieb zu einem Aufschaukeln des ALAADy-Demonstrators kommen, was zu ungewollten Flugzuständen wie zum Beispiel einem zu hohen Hängewinkel $\phi > 25\,^{\circ}$ führen kann. Im Rahmen dieser Arbeit ist t_{sec} auf einen Wert von 10 Sekunden festgelegt.
Zum Ende der While-Schleife wird die Bool-Variable `DAA_FLAG` an den Autopilot (AP) des ALAADy-Demonstrators übergeben. Die Bool-Variable `DAA_FLAG` gibt vor, ob der berechneten Guidance gefolgt werden soll. Wird die Bool-Variable `DAA_FLAG=false` übergeben, so folgt der AP des ALAADy-Demonstrators der originalen Flugplanung. Die Berechnete Guidance wird in jedem Zyklus an den AP übergeben und in der Funktion `command_guidance_from_array()` automatisch an die Flugleistungsgrenzen des ALAADy-Demonstrators angepasst. Die im Array `guidance[]` enthaltenen Werte sind auf die Aktualisierungsrate von DAIDALUS von $f = 1\,\mathrm{Hz}$ beschränkt. Da die in Zeile 61 aufgerufene Funktion allerdings in jedem Durchgang der While-Schleife und nicht nur ein Mal pro Sekunde aufgerufen wird, muss die `guidance[]` in der Funktion `command_guidance_from_array()` entsprechend heruntergebrochen werden.

3.2. Systemarchitektur

Dieses Unterkapitel behandelt die zur Erarbeitung der Ergebnisse verwendete Systemarchitektur. Abschnitt 3.2.1 bietet eine Vorstellung des vorhandenen ALAADy-MM und wie dieser mit auftretenden DAA Szenarien umgeht. In Abschnitt 3.2.2 wird die erstellte Kommunikationsschnittstelle zwischen DAIDALUS und dem ALAADy-Demonstrator mit seinen Subsystemen präsentiert. Abschließend werden in Abschnitt 3.2.3 einige Annahmen im Rahmen dieser Arbeit getroffen.

3.2.1. ALAADy-Missionsmanager

Der ALAADy-MM ist Teil des AP des ALAADy-Demonstrators. Während der AP dafür zuständig ist, die Geschwindigkeits- und Richtungsvorgaben umzusetzen, berechnet der MM unter anderem diese Vorgaben. Der MM ist als Zustandsautomat aufgebaut. Der Zustandsautomat beinhaltet diverse Zustände von der automatisierten Operation am Boden, über den automatisierten Flug, bis zur automatisierten Landung. Im Rahmen dieser Arbeit wird der Zustandsautomat um einen neuen Zustand (DAA Zustand) ergänzt. Der in Unterkapitel 3.1.2 beschriebene Code läuft kontinuierlich mit einer Aktualisierungsrate von $f = 1\,\mathrm{Hz}$. Sofern DAIDALUS eine Kollisionswarnung ausgibt (`IF DAIDALUS::alertLevelAllTraffic()>1`), wird der neu erstellte DAA Zustand aktiviert, nachdem eine neue Guidance durch DAIDALUS berechnet ist. Der DAA Zustand wird im ALAADy-MM mit Hilfe der Bool-Variable `DAA_FLAG` aktiviert.
Neben dem automatisierten Betrieb kann zu jeder Zeit auf den manuellen Betrieb umgeschaltet werden. So kann der Sicherheitspilot bei Flugversuchen in der Realität im Notfall immer die Kontrolle über den ALAADy-Demonstrator erhalten.
Im Rahmen dieser Arbeit werden die Zustände des automatisierten Fluges betrachtet. Hier stehen die Zustände *Global Trajectory*, *Local Trajectory* und *Loiter* zur Verfügung.

Abbildung 3.1 zeigt mögliche Flugführungen mit den genannten Zuständen.

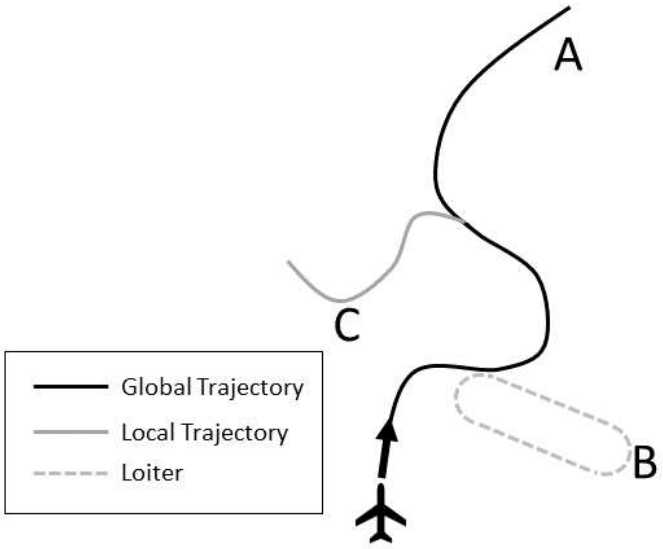


Abbildung 3.1 *Global Trajectory*, *Local Trajecotry* und *Loiter*

Im Zustand *Global Trajectory* wird ein zuvor eingespeicherter Flugpfad abgeflogen (vgl. A in Abbildung 3.1). Der Zustand kann zum Beispiel verlassen werden, um eine Warteschleife (in der Abbildung mit B markiert, Zustand *Loiter*) zu fliegen. Außerdem kann mit *Local Trajectory* (C) ein definierter Flugpfad von der aktuellen Position aus abgeflogen werden.
Aus jedem dieser drei Zustände wird im Falle einer Kollisionswarnung in den neu erstellten DAA Zustand gewechselt. Der DAA Zustand wird mit einer höheren Priorität behandelt als das Abfliegen des vorgegebenen Flugpfades in den drei Zuständen des automatisierten Fluges. Sobald die Kollisionswarnung verschwunden ist, d. h. das Ausweichmanöver ist abgeschlossen, wird der zuletzt berechnete Kurs für $\mathtt{t}_{sec}$ Sekunden weitergeflogen. Nach Ablauf der $\mathtt{t}_{sec}$ Sekunden wird wieder in den vorherigen Zustand gewechselt und der Flug wie gewünscht fortgesetzt.

3.2.2. DAL-Bridge

Die in dieser Arbeit präsentierte Implementierung des DAIDALUS Algorithmus wird in das bereits bestehende System des ALAADy-Demonstrators eingefügt. Ein korrektes Zusammenspiel der einzelnen Systemkomponenten ist daher zwingend erforderlich. Abbildung 3.2 zeigt schematisch das Zusammenspiel der einzelnen Systemkomponenten auf.

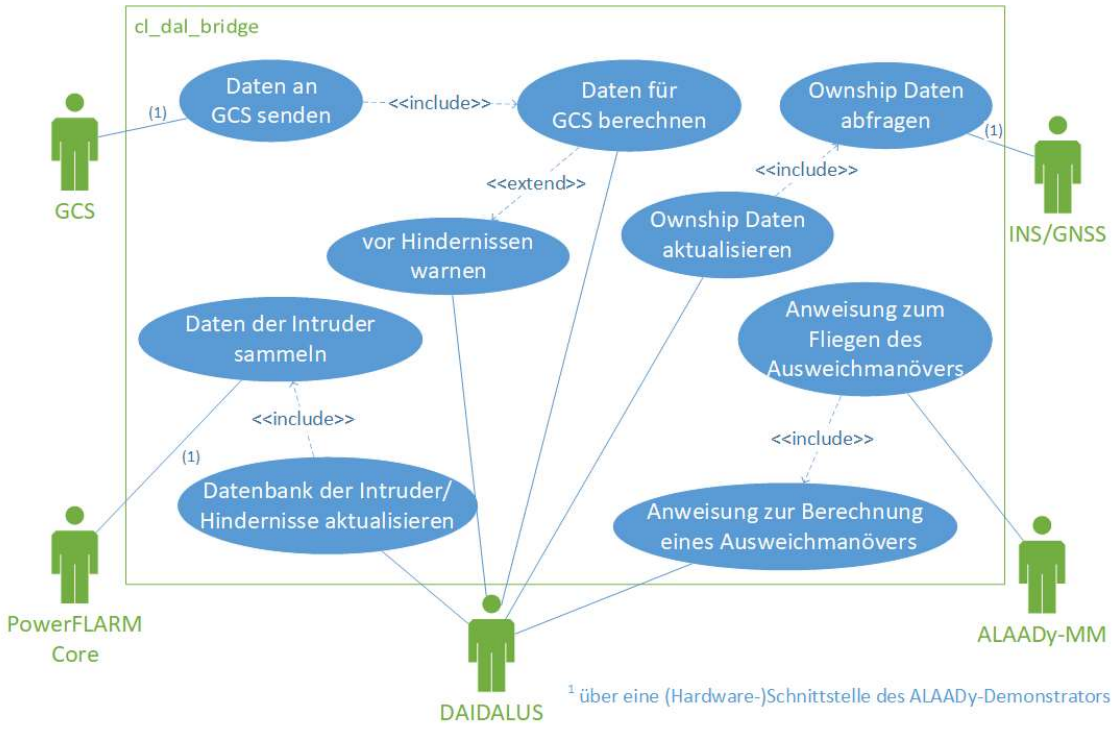


Abbildung 3.2 Use Case Diagramm

Wie in Abbildung 3.2 zu sehen ist, besteht das zu betrachtende System aus den Komponenten GCS, PowerFLARM Core, DAIDALUS, dem ALAADy-Missions Manager sowie einem INS/GNSS. Dabei sind die Komponenten PowerFLARM Core und INS/GNSS über Hardware-Komponenten, die über ein entsprechendes Interface im ALAADy-Demonstrator mit CIC und/oder FCC verbunden sind, realisiert. Die GCS befindet sich am Boden und erhält die nötigen Daten des ALAADy-Demonstrators über eine Funkverbindung. Bei den Komponenten DAIDALUS und ALAADy-MM handelt es sich um Softwarebausteine. Der im Zuge dieser Arbeit erstellte Code muss die fehlerfreie Interaktion der einzelnen Komponenten gewährleisten. Der geschriebene Code ist der Klasse `cl_dal_bridge` zuzuordnen. Dabei steht dal für DAIDALUS-ALAADy. Darüber hinaus ist die `cl_dal_bridge` dafür vorbereitet, der GCS die von DAIDALUS berechneten Bänder und Daten von erkannten Luftfahrzeugen/Hindernissen jeweils in einem struct bereitzustellen.
Die Kernaufgabe der `cl_dal_bridge` besteht in der Sicherstellung der Interaktion zwischen DAIDALUS und dem ALAADy-MissionsManager (ALAADy-MM). Dabei werden die Positions- und Geschwindigkeitsdaten des ALAADy-Demonstrators über die `cl_dal_bridge` an DAIDALUS übergeben. DAIDALUS nutzt diese Daten - gepaart mit den entsprechenden Daten der anderen bekannten Luftfahrzeuge - um vor möglichen Kollisionen zu warnen. Die Daten der anderen Luftfahrzeuge werden über das PowerFLARM Core ADS-B Modul empfangen. Das anschließend von DAIDALUS berechnete Ausweichmanöver wird dem ALAADy-MM als neue Soll-Daten vorgegeben.
Neben sich bewegenden fremden Luftfahrzeugen werden im Rahmen dieser Arbeit auch statische Hindernisse von der `cl_dal_bridge` an DAIDALUS übergeben. Dabei kann es sich zum Beispiel um Geofences handeln. Geofences sind virtuelle, meist durch mehrere

Positionspunkte aufgespannte, Bereiche. Sie können in der Höhe beschränkt sein. Es gibt Geofences, die das Ownship nicht verlassen darf. Diese Art des Geofence wird im Rahmen dieser Arbeit als *keep-in Geofence* bezeichnet. Außerdem gibt es Geofences, in die das Ownship nicht einfliegen darf. Diese zweite Art des Geofence wird im Rahmen dieser Arbeit als *keep-out Geofence* bezeichnet [52]. Die jeweiligen statischen Hindernisse werden mittels mehrerer virtuellen statischen Hindernissen nachgebildet. Abbildung 3.3 zeigt eine mögliche Nachbildung von verschiedenen statischen Hindernissen.

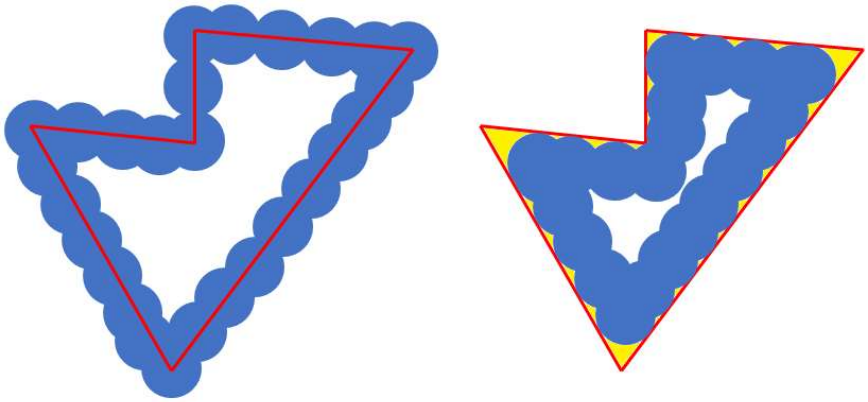


Abbildung 3.3 Nachbildung von statischen Hindernissen

Es bestehen verschiedene Möglichkeiten, die statischen Hindernisse durch mehrere virtuelle Hindernisse nachzubilden. In der Abbildung wird beispielhaft ein durch einen roten Umriss dargestellter Geofence nachgebildet. Die virtuellen Hindernisse (blau) können beispielsweise mittig auf dem Außenumriss liegen (wie in Abbildung 3.3 links gezeigt). Die virtuellen Hindernisse können aber beispielsweise auch tangential an der Außengrenze des originalen Hindernisses verlaufen. Durch die mittige Ausrichtung der virtuellen Hindernisse wird zwar mehr Platz eingenommen, als eigentlich gewollt. Allerdings wird dadurch das nachzubildende Hindernis komplett innerhalb der virtuellen Hindernisse gehalten. Bei einer Nachbildung, bei denen die virtuellen Hindernisse die Innenseite des statischen Hindernisses nur (von innen) tangential berühren, gibt es fälschlicherweise erlaubte Stellen innerhalb des statischen Hindernisses (in Abbildung 3.3 gelb markiert).
Eine wichtige Funktion, die von autonom fliegenden unbemannten Luftfahrzeugen beherrscht werden muss, ist das Abfliegen einer vorgegebenen Trajektorie. Im Rahmen dieser Arbeit wird ein zweidimensionaler Flugpfad durch statische Hindernisse nachgebildet. Dabei dürfen die statischen Hindernisse, die zur Nachbildung des Flugpfades genutzt werden, nicht direkt auf dem Flugpfad liegen. Vielmehr muss der Flugpfad von statischen Hindernissen umgeben sein. So entsteht ein Korridor, in dem sich das Ownship bewegen darf. Der Korridor sollte dabei so breit sein, dass das Ownship problemlos darin fliegen kann, ohne die Volumen der statischen Hindernisse zu verletzen. Andererseits muss der Korridor eng genug sein, sodass das Ownship nicht wenden kann und in die entgegengesetzte Richtung fliegt. Der Korridor um die geplante Trajektorie wird nicht als optimale Lösung zum Abfliegen einer Trajektorie geplant. Vielmehr lässt sich damit eine Rückfallebene erstellen, falls die primäre Methode zum Abfliegen der Trajektorie ausfällt. So kann das Ownship weiterhin innerhalb eines geplanten

Bereichs operieren.
Auch die Größe der zur Nachbildung verwendeten virtuellen Hindernisse kann durch die Zuordnung unterschiedlicher Alerter variiert werden. Generell ist dabei zu beachten, dass eine höhere Anzahl an virtuellen Hindernissen zu einer komplexeren Berechnungsgrundlage für DAIDALUS führt. Zu groß gewählte Dimensionen können die exakte Nachbildung von komplexen Strukturen erschweren. Einzelne zylindrische Hindernisse können durch ein einziges Hindernis in DAIDALUS repräsentiert werden. Die Größe des Hindernisses lässt sich durch die in Tabelle 2.1 aufgeführten Parameter an die benötigte Größe anpassen.

3.2.3. Annahmen im Rahmen der Arbeit

Für eine einfachere Betrachtung der Szenarien wurden, sofern in den einzelnen Abschnitten nicht anders beschrieben, für die gesamte Bearbeitung dieser Arbeit folgende Annahmen getroffen:

- Für die Berechnungen wird eine (lokal) flache Erde angenommen. Sowohl die Erdkrümmung als auch mögliche Geländeerhebungen werden nicht beachtet.
- Die Position und Bewegung der Luftfahrzeuge wird innerhalb eines erdlotfestes kartesisches Koordinatensystem beschrieben. Die z_g-Achse zeigt dabei entlang des Gewichtsvektors senkrecht nach unten. Die x_g-Achse zeigt in Richtung Norden, die y_g-Achse zeigt entsprechend in Richtung Osten. Das erdlotfeste kartesische Koordinatensystem wird auch als *North East Down* (NED)-Koordinatensystem bezeichnet. Ein Kurs von $\chi = 360\,^\circ$ ist parallel zur x_g-Achse definiert, ein Kurs von $\chi = 90\,^\circ$ verläuft parallel zur y_g-Achse.
- Die Positions- und Geschwindigkeitsvektoren zur Bestimmung des Flugzustandes eines Luftfahrzeugs im oben definierten erdlotfesten Koordinatensystem sind wie folgt definiert:

$$\vec{x}_g = \begin{pmatrix} x_g \\ y_g \\ z_g \end{pmatrix}, \quad \vec{v} = \dot{\vec{x}}_g = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

- Des Weiteren wird für alle Berechnungen Windstille angenommen. Dadurch ist die Funktion von DAIDALUS unabhängig von äußeren Einflüssen analysierbar.
- DAIDALUS bietet die Möglichkeit über Änderungen des Kurses sowie über horizontale und vertikale Geschwindigkeitsänderungen möglichen Hindernissen auszuweichen. Dadurch ist eine differenzierte Betrachtung der gesammelten Daten möglich. Mögliche Einflüsse einer Änderung von Vertikal- oder Horizontalgeschwindigkeit auf die Ergebnisse werden so ausgeschlossen.
- Anwendungsfall dieser Arbeit ist der in Unterkapitel 1.3 vorgestellte ALAADy-Demonstrator des DLRs in Braunschweig. Daher sollen alle Ausweichmanöver von diesem Luftfahrzeug geflogen werden können. Die Flugleistungsgrenzen des ALAADy-Demonstrators sind un-

bedingt einzuhalten. Die Flugleistungsgrenzen sind im Anschluss an diese Auflistung in Tabelle 3.1 aufgeführt. Für die Betrachtungen in dieser Arbeit wird eine konstante Fluggeschwindigkeit von $v_A = 25\,m/s = 90\,km/h$ angenommen. Die Fluggeschwindigkeit entspricht der im AP genutzten Reisegeschwindigkeit.

- Im Realbetrieb des ALAADy-Demonstrators wird eine Aktualisierungsrate der FLARM Daten von 1Hz erwartet. Daher wird die in Abschnitt 3.1.2 beschriebene DAIDALUS Funktion mit 1Hz ausgeführt. Ein häufiger Aufruf der Berechnung durch DAIDALUS mit denselben Werten erhöht die Auslastung des Systems unnötig.

Tabelle 3.1 Flugleistungsgrenzen ALAADy-Demonstrator

Parameter	Wert
Fluggeschwindigkeit v_A	$90\,\mathrm{km/h} = 25\,\mathrm{m/s} (const)$
max. Hängewinkel ϕ	$25\,^\circ$
max. Steiggeschwindigkeit[2] w	$-5\,\mathrm{m/s}$
max. Sinkgeschwindigkeit[2] w	$+4\,\mathrm{m/s}$

Der minimale Wenderadius eines Luftfahrzeugs kann mit folgender Formel berechnet werden:

$$r_{w,min} = \frac{v_A^2}{g * \tan(\phi)} \tag{3.1}$$

Durch die oben gegebenen Flugleistungsgrenzen ergibt sich unter der Verwendung der Erdbeschleunigung $g = 9,81\,\mathrm{m/s^2}$ für den ALAADy-Demonstrator ein minimaler Wenderadius von $r_{w,min} = 136,6\,\mathrm{m}$.

3.3. Simulationsumgebungen

In diesem Kapitel werden die im Rahmen dieser Arbeit verwendeten Simulationsmodelle vorgestellt. Abschnitt 3.3.1 präsentiert das laufzeiteffiziente kinematische Bewegungsmodell. In Abschnitt 3.3.2 wird das dynamische Bewegungsmodell vorgestellt.

3.3.1. Kinematisches Bewegungsmodell

Um eine laufzeiteffiziente Berechnung von DAA Szenarien gewährleisten zu können, wird für diese Arbeit eine Simulationsumgebung mit einem kinematischen Bewegungsmodell programmiert. Dieses Bewegungsmodell ist in einer eigenen Klasse hinterlegt. Das Bewegungsmodell orientiert sich dabei an einer Punktmasse und berechnet die aktuelle Position auf Basis der vorherigen Position sowie dem gegebenen Geschwindigkeitsvektor und der seit

[2] Die Steiggeschwindigkeit ist im erdlotfestes kartesisches Koordinatensystem negativ, die Sinkgeschwindigkeit positiv definiert

der letzten Berechnung vergangenen Zeitdifferenz:

$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_t * \Delta t \tag{3.2}$$

Zur schnellen und einfachen Verifizierung wird eine einfache Visualisierung genutzt. Darin wird jedes Luftfahrzeug (auch fremde Luftfahrzeuge) als Pfeil mit der eigenen Position bezogen auf ein inertiales NED-Koordinatensystem dargestellt. Ebenfalls dargestellt wird die Lage im Raum anhand von Bahnazimut-, Bahnneigungs- und Hängewinkel. Die Winkel werden auf Basis des aktuellen Geschwindigkeitsvektors berechnet. Da für die Simulation kein Wind angenommen wird, werden die Winkel wie folgt berechnet:

$$\text{Bahnazimutwinkel: } \chi = \begin{cases} -\arcsin\left(\frac{v}{\sqrt{u^2+v^2}}\right) + \pi & \text{für } u < 0 \\ \arcsin\left(\frac{v}{\sqrt{u^2+v^2}}\right) & \text{für } u \geq 0 \end{cases} \tag{3.3}$$

$$\text{Bahnneigungswinkel: } \gamma = -\arcsin\left(\frac{w}{\sqrt{u^2+v^2+w^2}}\right) \tag{3.4}$$

$$\text{Hängewinkel: } \phi = 0 \tag{3.5}$$

Die Winkel sind jeweils rechtsdrehend von der entsprechenden Achse definiert. Der Bahnazimutwinkel ist im Intervall $\chi \in [0; 2\pi)$ definiert mit 0 in Richtung Norden. Der Bahnneigungswinkel ist im Intervall $\gamma \in [-\pi; +\pi)$ definiert. Negative Werte entsprechen dabei einem *Nose down*-, positive einem *Nose up*-Zustand. Mit den in Abschnitt 3.2.3 definierten Vereinfachungen entspricht der Bahnazimutwinkel dem Kurs des Ownships. Damit kann der Bahnazimutwinkel als direkter Eingabewert für DAIDALUS genutzt werden. Außerdem kann der Bahnneigungswinkel mit den in Abschnitt 3.2.3 definierten Vereinfachungen zur Berechnung der Vertikalgeschwindigkeit genutzt werden. Dabei gilt:

$$\dot{z} = w = \sin\gamma * ||\vec{v}|| \tag{3.6}$$

Der Hängewinkel ϕ wird von DAIDALUS nicht benötigt und hat keinen Mehrwert bei Verwendung des kinematischen Modells. Er ist mit 0 definiert und hier lediglich der Vollständigkeit wegen aufgeführt. Die oben beschriebene Simulationsumgebung wird innerhalb dieser Arbeit als *kinematische Simulation* bezeichnet.

3.3.2. Dynamisches Bewegungsmodell

Nachdem die Funktionsweise der `cl_dal_bridge` in der Simulation mit dem kinematischen Bewegungsmodell verifiziert ist, wird die `cl_dal_bridge` auch in der vom DLR entwickelten *Software In The Loop* (SITL) Simulation überprüft. Im Gegensatz zu der Simulation mit dem vereinfachten Bewegungsmodell, nutzt die hier verwendete Simulationsumgebung ein

detailliertes flugdynamisches Modell des ALAADy-Demonstrator. Die Berechnungen der einzelnen Szenarien benötigen daher mehr Rechenzeit, liefern aber realistischere Ergebnisse. Fremde Luftfahrzeuge werden in der Simulationsumgebung des flugdynamischen ALAADy-Bewegungsmodell mit einem konstanten Geschwindigkeitsvektor und unter Verwendung des in Abschnitt 3.3.1 beschriebenen kinematischen Bewegungsmodells simuliert.
Da die Nutzung der Simulationsumgebung mit dem flugdynamischen Modell deutlich rechenintensiver ist, werden die Ergebnisse dieser Arbeit mit der kinematischen Simulation berechnet. Einzelne Szenarien werden in der Simulation mit dem flugdynamischen Modell nachgestellt und sollen so die Ergebnisse aus der kinematischen Simulation verifizieren. Die Simulationsumgebung mit dem flugdynamischen Modell wird im Rahmen dieser Arbeit als *dynamische Simulation* bezeichnet.

4. Ergebnisse und Diskussion

In diesem Kapitel werden die im Rahmen dieser Arbeit erarbeiteten Ergebnisse dargestellt und diskutiert. In Unterkapitel 4.1 werden Szenarien zum Ausweichen gegenüber statischen Hindernissen betrachtet. Anschließend werden in Unterkapitel 4.2 sich bewegende Luftfahrzeuge als Hindernisse betrachtet. In Unterkapitel 4.3 werden statische und sich bewegende Hindernisse in kombinierten Szenarien betrachtet. Die präsentierten Daten stammen aus Berechnungen der kinematischen Simulation. Sofern die Daten aus der dynamischen Simulation stammen, ist dies explizit angegeben.
Alle Berechnungen wurden unter Verwendung der Version DAIDALUS 2.0.2c durchgeführt. Als Hardware wurde ein Windows 10 Computer mit folgenden Hardware Spezifikationen genutzt:

Tabelle 4.1 Verwendete Hardware

Komponente	Daten
Prozessor	Intel Core i7-6600U @ 2.6GHz
Arbeitsspeicher	16GB DDR4
Festplattenspeicher	512 GB SATA III-SSD

4.1. Statische Hindernisse

In den folgenden Abschnitten werden Szenarien mit statischen Hindernissen betrachtet. Sofern nicht anders angegeben, werden die Hindernisse durch den in der `DO_365A_SUM.conf` [58] hinterlegten Alerter *DWC_Phase_I_SUM* (*Standard Alerter*) repräsentiert. Der Standard Alerter entspricht den in der DO-365 definierten Werten. Die Werte sind in Tabelle 2.1 aufgeführt.
In Abschnitt 4.1.1 wird der Einfluss der Position des Ownships bei Erkennung der Hindernisse sowie die allgemeine Funktion des Algorithmus anhand von zwei statischen Hindernissen untersucht. Anschließend wird in Abschnitt 4.1.2 betrachtet, inwiefern der Abstand zweier statischer Hindernisse zueinander das Ergebnis beeinflusst. In Abschnitt 4.1.3 wird beurteilt, wie zuverlässig Geofences mit dem DAIDALUS Algorithmus eingehalten werden. Im letzten Abschnitt dieses Unterkapitels (4.1.4) wird evaluiert, ob sich das Ownship in einem Korridor aus Hindernissen halten kann. Alle Szenarien werden - sofern nicht anders beschrieben - unter Berücksichtigung der folgenden Parameter durchgeführt:

Tabelle 4.2 Verwendete Flugzustandsgrößen

Parameter	Wert
Fluggeschwindigkeit Ownship v_A	25 m/s (90 km/h)
Flughöhe Ownship h_O	100 m AGL[1]
Höhe (Mittelpunkt) statische Hindernisse h_H	100 m AGL[1]
Erlaubte Ausweichrichtung	horizontale Kursänderung

4.1.1. Zwei statische Hindernisse

Mit diesem Szenario soll die grundlegende Funktion des DAIDALUS Algorithmus analysiert werden. So kann überprüft werden, ob die Hindernisse von DAIDALUS mit den korrekten Koordinaten und Parametern übernommen werden. Bei Verwendung des *Standard-Alerters* beträgt der Radius des *Corrective Volumes* ca. 1220 m (s. Tabelle 2.1). Um ein (frei definiertes) Hindernis nachzuahmen, sind die Mittelpunkte der Hindernisse $\Delta y_g = 1000\,\mathrm{m}$ von einander entfernt. Dadurch überschneiden sich die *Corrective Volumes* der Hindernisse gegenseitig.
Für die Auswertung werden die Hindernisse vom Ownship aus verschiedenen Positionen und mit verschiedenen Kursen angeflogen. Abbildung 4.1 bietet eine schematische Darstellung des Szenarios.

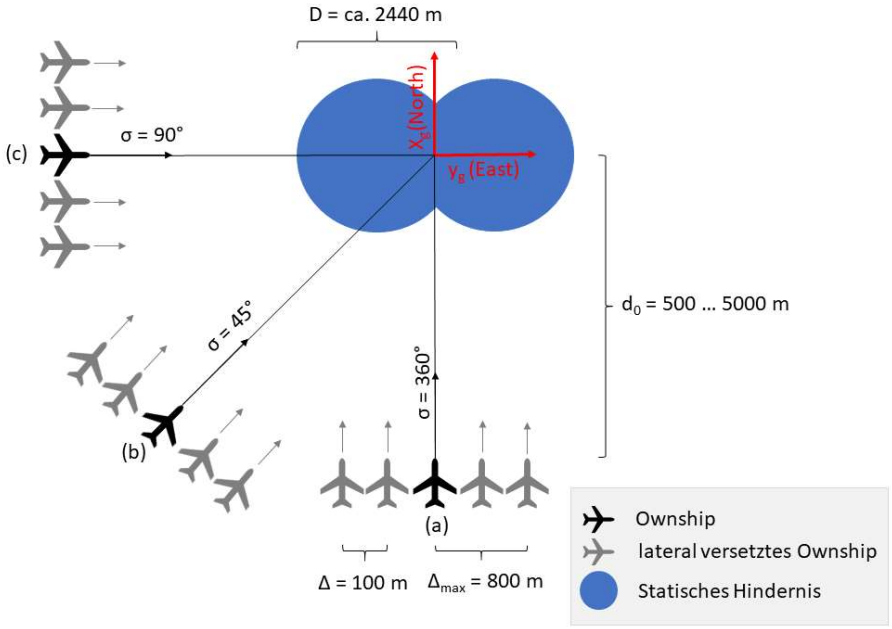


Abbildung 4.1 Schematische Darstellung Szenario zwei statische Hindernisse

Das Ownship startet mit Kurs $\chi = 360\,^\circ$ (a), mit Kurs $\chi = 90\,^\circ$ (c) sowie aus mit Kurs $\chi = 45\,^\circ$ (b). Die Distanz des Ownships zum Koordinatenursprung wird dabei in 250 m-Schritten von

[1] entspricht einer z_g-Koordinate von -100

$d_0 = 500\,\mathrm{m}$ auf $d_0 = 5000\,\mathrm{m}$ erhöht. Des Weiteren wird für jede Distanz und Flugrichtung die Abflugposition in Schritten von $\Delta = 100\,\mathrm{m}$ bis maximal $\Delta_{max} = 800\,\mathrm{m}$ variiert.

Fall (a): Anflug mit Kurs $\chi = 360\,^\circ$

Zu Beginn dieses Abschnittes wird das Szenario eines Anfluges aus einer Distanz von $d_0 = 5000\,\mathrm{m}$ betrachtet. Das Szenario bietet DAIDALUS die längste Zeitspanne innerhalb der Versuchsreihe, um ein Ausweichmanöver zu berechnen und ein Einfliegen in die *Separation Volumes* zu verhindern. Abbildung 4.2 zeigt einen Plot aller 17 Flüge aus 5000 m Entfernung. Jede einzelne Linie beschreibt einen separaten Flug. Die blau eingefärbten Kreise repräsentieren jeweils ein statisches Hindernis. Die Radien der blauen Kreise betragen entsprechend des verwendeten *Standard-Alerters* $DTHR = 1220\,\mathrm{m}$. Die roten Kreise repräsentieren das *NMAC Volume* mit einem Radius von $DTHR_{NMAC} = 500\,\mathrm{m}$.

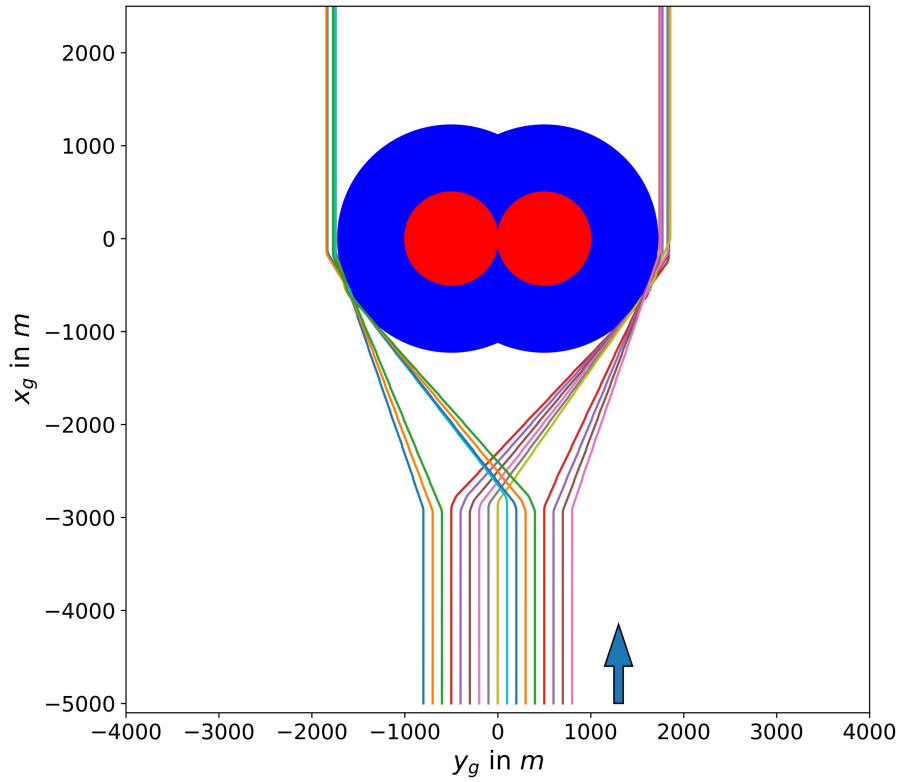


Abbildung 4.2 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 360\,^\circ$ aus $d_0 = 5000\,\mathrm{m}$ Entfernung

Wie in jedem Szenario dieses Abschnittes liegen die Mittelpunkte der beiden statischen Hindernisse bei $y_g = \pm 500\,\mathrm{m}$ auf der y_g-Achse ($x_g = 0\,\mathrm{m}$). Der Startpunkt jedes einzelnen Fluges liegt bei $x_g = -5000\,\mathrm{m}$. Die Abflugposition variiert dabei zwischen $y_g = -800\,\mathrm{m}$ und $y_g = +800\,\mathrm{m}$. Der blaue Pfeil gibt den Kurs $\chi = 360\,^\circ$ zu Beginn des Fluges an.
Wie in Abbildung 4.2 zu sehen ist, behalten alle Flüge den Kurs von $\chi = 360\,^\circ$ bei, bis ihr Abstand zu den Mittelpunkten der statischen Hindernisse auf etwa $d = 3000\,\mathrm{m}$. Der Anteil des Geradeausfluges wird bei den Fällen mit weniger als 5000 m Entfernung entsprechend verkürzt. Durch Auflösen von Gleichung 2.4 ($\tau_{mod} = \frac{-(r^2 - DMOD^2)}{r\dot{r}}$) nach r kann die Distanz, bei der die horizontale Zeitkonstante unterschritten ist, berechnet werden. Bei einer Distanz von

$d = 2957,9\,\mathrm{m}$ ist die horizontale Zeitkonstante $TTHR$ des *Corrective Volumes* unterschritten und es wird ein *Corrective Alert* (`DAIDALUS::alertLevelAllTraffic()=2`) von DAIDALUS ausgegeben. Daraus resultiert die Aktivierung der `DAA_FLAG` und die Berechnung eines Ausweichmanövers.
Betrachtet man das Ausweichmanöver selbst, so würde ein menschlicher Pilot je nach Position entweder links oder rechts an den Hindernissen vorbeifliegen. Die von DAIDALUS berechneten Ausweichmanöver spiegeln dieses Verhalten nur bedingt wider. Die jeweils äußeren 3-4 Flüge nehmen wie erwartet den kürzesten Weg um die Hindernisse. Bei Betrachtung der mittleren Flüge fällt allerdings auf, dass sich deren Flugbahnen überschneiden. Die Ausweichmanöver der Flüge mit einer Startposition von $y_g = -100\,\mathrm{m}$ bis $y_g = -500\,\mathrm{m}$ führen entgegen der Erwartung rechts an beiden Hindernissen vorbei. Auch der mittig gestartete Flug passiert die Hindernisse auf der rechten Seite. Die Flüge mit einer Startposition von $y_g = 100\,\mathrm{m}$ bis $y_g = 400\,\mathrm{m}$ umfliegen die Hindernisse auf der linken Seite. Zu erklären ist dieses für den Menschen unerwartete Ausweichverhalten durch den Algorithmus von DAIDALUS. Der Algorithmus sucht nach dem nächstgelegenen Kurs, der zu einer möglichst geringen Aufenthaltszeit im zu vermeidenden Volumen führt. Anstatt auf der zu erwartenden Seite an den Hindernissen vorbeizufliegen, wird der Kurs in Richtung des abnehmenden Radius des jeweils unmittelbar vor dem Ownship liegenden Hindernisses korrigiert. Das zweite Hindernis führt zu diesem Zeitpunkt noch nicht zum Auslösen eines *Corrective Alerts* durch DAIDALUS, da die horizontale Zeitkonstante $TTHR$ noch nicht unterschritten ist. Kurz nach Einleiten des Ausweichmanövers sorgt allerdings auch das zweite Hindernis zum Auslösen eines *Corrective Alerts*. Das nun kürzeste Ausweichmanöver besteht in der weiteren Ausführung des bereits eingeleiteten Ausweichmanövers. Die unterschiedlichen y_g -Werte nach Passieren der Hindernisse ist durch die Rückführung auf den ursprünglich geflogenen Kurs (vgl. Zeile 45 im Codebeispiel in Abschnitt 3.1.2) zu erklären. Der ursprüngliche Kurs von $\chi = 360\,^\circ$ wird erst eingeschlagen, wenn `IF (current_time - alert_since) > t_sec` wahr ist. Im Rahmen dieser Arbeit ist `t_sec` mit 10 Sekunden definiert. Weil die Ausweichmanöver zu unterschiedlichen Zeitpunkten und an unterschiedlichen Positionen eingeleitet werden, weichen auch die Positionen, bei denen oben genannte Bedingung wahr wird, geringfügig voneinander ab.
Wie in Abbildung 4.2 zu sehen ist, weicht jeder einzelne Flug erfolgreich den statischen Hindernissen aus. Der Abstand zu den Mittelpunkten der statischen Hindernisse beträgt zu jedem Zeitpunkt mindestens $d = 1220\,\mathrm{m}$. Eine in Abbildung 4.2 als Überschneidung erscheinende Linie ist der Bildauflösung des Plots und der im Plot verwendeten Linienbreite der Flüge geschuldet.
Bei einer Anfangsdistanz von $d_0 = 2750\,\mathrm{m}$ weichen, wie in Abbildung 4.3 zu sehen, alle berechneten Flüge wie von einem menschlichen Betrachter intuitiv erwartet in die richtige Richtung aus. Hier ist die oben berechnete kritische Distanz zum Auslösen eines *Corrective Alerts* bereits zu Beginn der Simulation gegeben und beide statischen Hindernisse werden von Beginn an von DAIDALUS beachtet. Damit wird das kürzeste Ausweichmanöver um beide statischen Hindernisse so berechnet, wie es die menschliche Intuition erwartet.

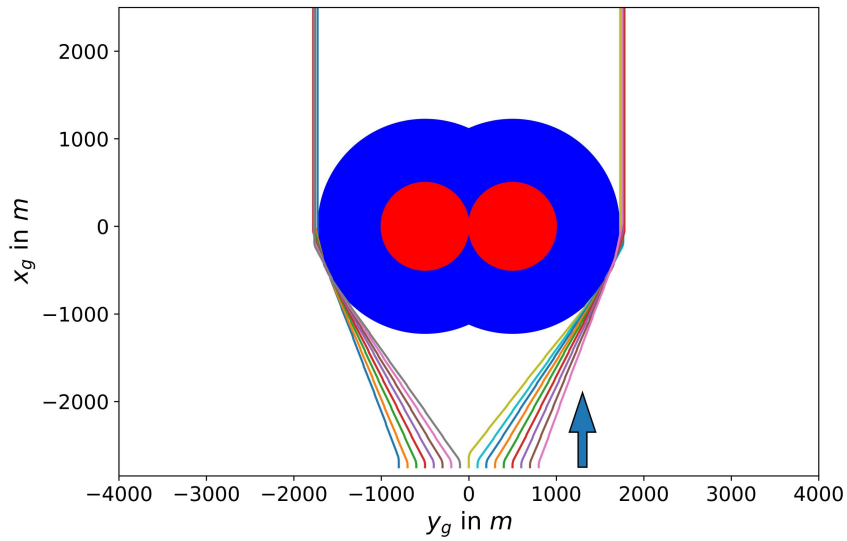


Abbildung 4.3 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 360\,°$ aus $d_0 = 2750\,\mathrm{m}$ Entfernung

Bei diesem Szenario kommt es weiterhin zu keiner Kollision mit den *Spearation Volumes* der statischen Hindernisse. Bis zu einer Angsngsdistanz von $d_0 = 1250\,\mathrm{m}$ verletzt keiner der Flüge die *Separation Volumes* der statischen Hindernisse. Erst bei einer Anfangsdistanz von $d_0 = 1250\,\mathrm{m}$ oder weniger wird das *Corrective Volume* von jedem getesteten Flug verletzt. Beispielhaft zeigt Abbildung 4.4 das Szenario mit einer Anfangsdistanz von $d_0 = 1250\,\mathrm{m}$.

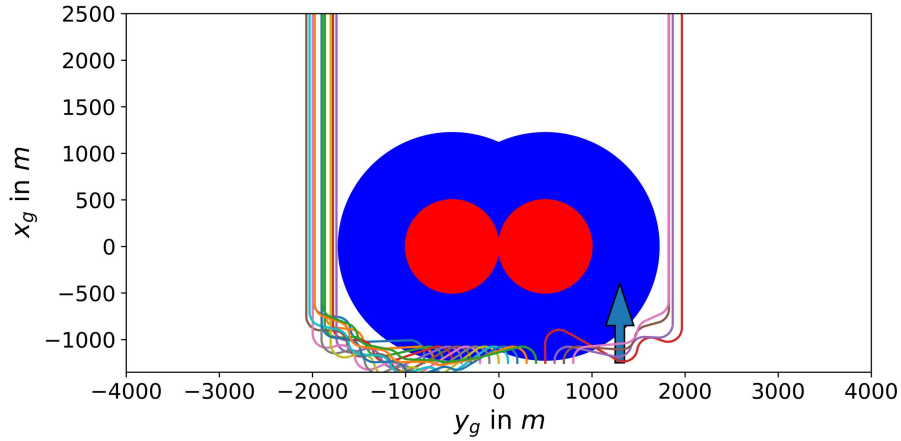


Abbildung 4.4 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 360\,°$ aus $d_0 = 1250\,\mathrm{m}$ Entfernung

Zu beachten ist hierbei, dass bei einer Startposition $x_g = -1250\,\mathrm{m}$ nur wenige Sekunden bis zum Eintritt in das *Corrective Volume* bleiben. Durch den Einschnitt, den die beiden Kreise der Hindernisse bilden, bleibt dem mittig startenden Flug ($y_g = 0\,\mathrm{m}$) in diesem Testfall die längste Strecke bis zum Eintritt in das *Corrective Volume*. Die Strecke bis zum Eintritt liegt dabei bei $137,2\,\mathrm{m}$, was einer Flugdauer von $5,5\,\mathrm{s}$ entspricht. Der minimale Wenderadius liegt wie mit Gleichung 3.1 berechnet bei $r_{w,min} = 136,6\,\mathrm{m}$. Die Strecke ist damit nur minimal länger als der minimale Wenderadius des ALAADy-Demonstrators. Aufgrund der Aktualisierungsrate von DAIDALUS von $f = 1\,\mathrm{Hz}$ wird die berechnete Ausweichempfehlung erst bei $t = 1\,\mathrm{s}$

umgesetzt. In dieser Zeit fliegt das Ownship bereits $v_A * t = 25\,\mathrm{m/s} * 1\,\mathrm{s} = 25\,\mathrm{m}$ geflogen. Eine Verletzung des *Corrective Volumes* ist damit nicht zu verhindern. Der bei $y_g = 500\,\mathrm{m}$ startende Flug kommt den Mittelpunkten auf eine minimale Distanz von $d = 906,6\,\mathrm{m}$ zum rechten Hindernis am nächsten. Bei diesem Flug gibt DAIDALUS zu Beginn des Fluges einen fehlerhaften Wert für die Kurskorrektur aus. Der fehlerhafte Wert liegt außerhalb des Wertebereichs für den Kurs χ und führt dadurch zu einer Verzögerung der Ausführung des Ausweichmanövers. Eine derartige Ausgabe kann laut einem der Mitentwickler von DAIDALUS in Einzelfällen auftreten [29]. Eine *NMAC* tritt erst bei einer Anfangsdistanz von $d_0 = 500\,\mathrm{m}$ auf. Die minimale Distanz liegt hierbei bei $d = 328,8\,\mathrm{m}$.
Bei der Nachbildung von Fall (a) dieses Abschnittes mit der dynamischen Simulation kommt es wie erwartet zu einem etwas anderen Verhalten des Ownships. Weil sich mit Einleiten einer Kurve der Auftriebsvektor durch die Schräglage des Ownships zur Seite neigt, kommt es zu einem geringeren Anteil des Auftriebsvektors in negativer z_g-Richtung. Daraus resultiert eine zwischenzeitliche Abnahme der Flughöhe um etwa $20\,\mathrm{m}$. Nachdem der Autopilot des ALAADy-Demonstrators den Bahneigungswinkel γ angepasst hat, wird die ursprüngliche Flughöhe wieder erreicht. Trotz des unterschiedlichen Bewegungsmodells kommt es auch in der dynamischen Simulation zu keiner Verletzung der *Separation Volumes* wenn die Anfangsdistanz von $d_0 \geq 1500\,\mathrm{m}$. Bei einer Anfangsdistanz von $d_0 = 1250\,\mathrm{m}$ kommt es zu einer Verletzung des *Corrective Volumes*. Das Bewegungsmodell der dynamischen Simulation leitet eine Kurve mit einer steigenden Schräglage des Ownships ein. In der kinematischen Simulation wird direkt in den stationären Kurvenflug gewechselt. Daraus resultiert ein etwas größerer Kurvenradius des Ownships in der dynamischen Simulation und die Verletzung des *Corrective Volumes*.

Fall (b): Anflug mit Kurs $\chi = 45\,^\circ$

Für Anfangsdistanzen von $d_0 = 5000\,\mathrm{m}$ bis $d_0 = 3000\,\mathrm{m}$ sind die Ergebnisse wie auch in Fall (a) untereinander nahezu identisch. Lediglich die Länge des Geradeausfluges (Kurs $\chi = 45\,^\circ$) wird mit sinkender Distanz entsprechend kürzer. Abbildung 4.5 zeigt die Flugpfade für das Szenario mit einer Anfangsdistanz von $d_0 = 5000\,\mathrm{m}$.

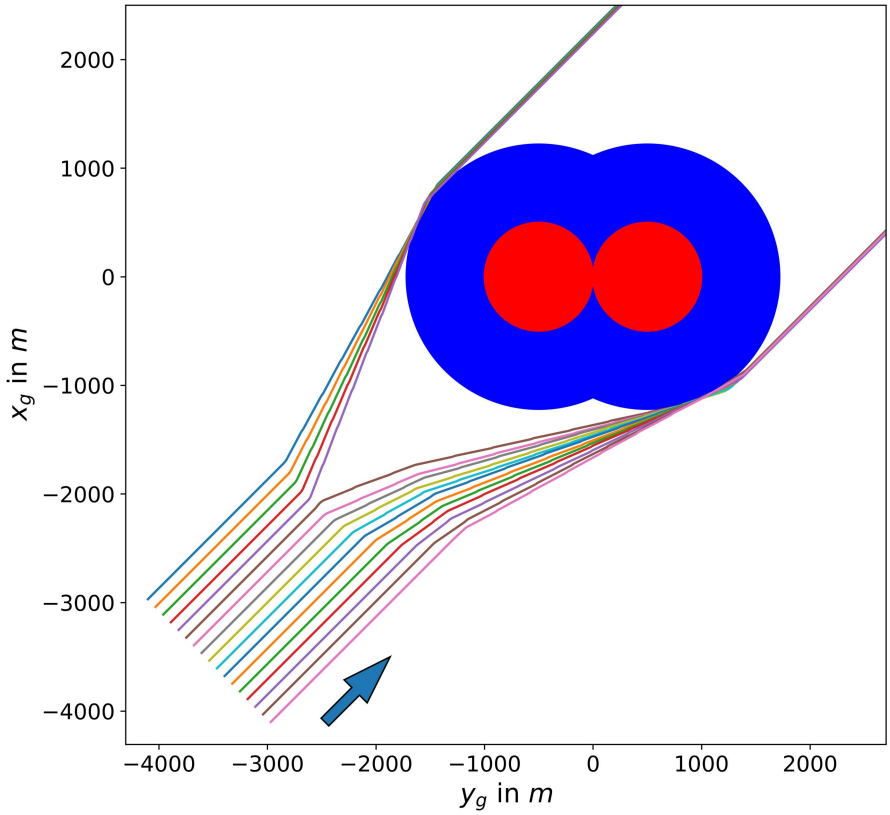


Abbildung 4.5 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 45\,^\circ$ aus $d_0 = 5000\,\mathrm{m}$ Entfernung

Wie in Fall (a) weichen alle simulierten Flüge bei einer Anfangsdistanz von $d_0 = 5000\,\mathrm{m}$ bis $d_0 = 3000\,\mathrm{m}$ den statischen Hindernissen aus und meiden die *Separation Volumes* der Hindernisse. Auffällig im Vergleich zu Fall (a) ist die deutliche Aufteilung in Ausweichmanöver, die rechts beziehungsweise links an den Hindernissen vorbeiführen. Eine Kreuzung der mittleren Flüge bleibt in Fall (b) aus.

Alle simulierten Flüge von einer Anfangsdistanz $d_0 \geq 2000\,\mathrm{m}$ verletzen keines der *Separation Volumes*. Bei einer Anfangsdistanz von $d_0 = 1750\,\mathrm{m}$ wird das *Corrective Volume* von zwei Flügen verletzt. Die minimale Distanz beträgt dabei $d = 1153\,\mathrm{m}$. Dabei gibt DAIDALUS zu Beginn des Fluges als erste Richtungsangabe für das Ausweichmanöver einen fehlerhaften Wert aus. Dadurch verzögert sich wie in Fall (a) die Ausführung des Ausweichmanövers und das *Separation Volume* wird verletzt.

Bei einer Anfangsdistanz von $d_0 = 1500\,\mathrm{m}$ oder weniger kreuzen die meisten simulierten Flüge mit einem Kurs zu Beginn von $\chi = 45\,^\circ$ das *Corrective Volume* der statischen Hindernisse. Abbildung 4.6 zeigt dieses Szenario. Dabei ist zu beachten, dass die meisten Flüge bereits im *Separation Volume* des statischen Hindernisses gestartet werden.

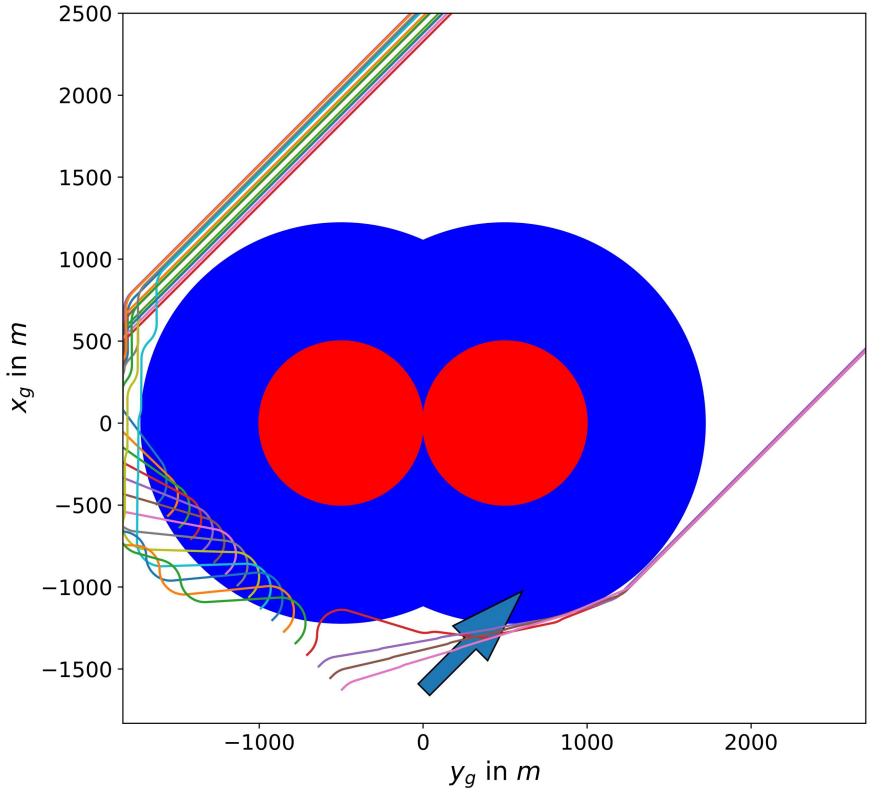


Abbildung 4.6 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 45\,^\circ$ aus $d_0 = 1500\,\mathrm{m}$ Entfernung

In Abbildung 4.6 fällt vor allem ein Flug aus den Mustern der anderen. Der Flug mit einer lateralen Verschiebung von $+500\,\mathrm{m}$ fliegt zu Beginn eine Linkskurve und verletzt so das Corrective Volume des linken statischen Hindernisses. Anschließend wird der Kurs durch eine Rechtskurve korrigiert, das Volumen wieder verlassen und rechts um beide Hindernisse geflogen. Die erste Linkskurve ist durch einen fehlerhaften ersten Wert für die Korrektur des Kurses zu erklären. Der Wert liegt außerhalb des Wertebereichs für den Kurs χ.
Die ersten *NMACs* treten bei einer Anfangsdistanz von $d_0 = 1000\,\mathrm{m}$ auf. Die minimale Distanz zum Mittelpunkt der statischen Hindernisse beträgt dabei $d = 472,6\,\mathrm{m}$, wobei dieser Flug zu Beginn eine Distanz von lediglich $d = 662\,\mathrm{m}$ beträgt.

Fall (c): Anflug mit Kurs $\chi = 90\,^\circ$

Im betrachteten Fall (c) starten die Flüge links von den zwei statischen Hindernissen mit einem Kurs von $\chi = 90\,^\circ$. Dabei ist zu beachten, dass sich die Anflugdistanz auf die Distanz zum Ursprung des Koordinatensystems (der zwischen den beiden statischen Hindernissen liegt) bezieht. Der Mittelpunkt eines der Hindernisse liegt $500\,\mathrm{m}$ links vom Ursprung des Koordinatensystems. Dadurch verringert sich die tatsächliche Anfangsdistanz ebenfalls um $500\,\mathrm{m}$. Die im Folgenden angegebenen Anfangsdistanzen beschreiben den y_g-Abstand zum Ursprung des Koordinatensystems. Abbildung 4.7 zeigt das Szenario des Anflugs mit Anfangsdistanz $d_0 = 5000\,\mathrm{m}$ und einem Kurs von $\chi = 90\,^\circ$.

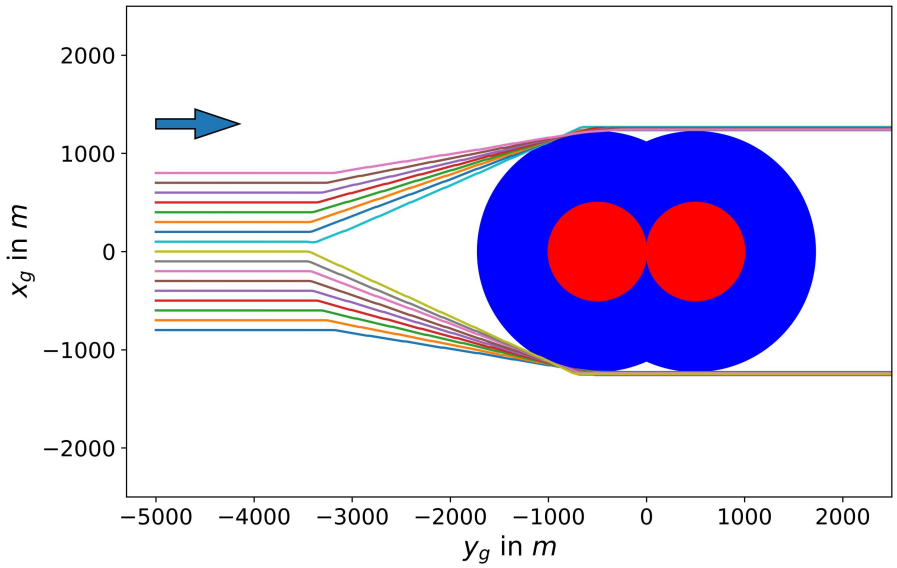


Abbildung 4.7 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 90\,^\circ$ aus $d_0 = 5000\,\mathrm{m}$ Entfernung

Wie auch in den Fällen (a) und (b) unterscheiden sich die Szenarien mit einer Anfangsdistanz von $d_0 = 5000\,\mathrm{m}$ bis $d_0 = 3500\,\mathrm{m}$ lediglich in der Länge des Geradeausfluges vor den statischen Hindernissen. In diesen Szenarien weichen alle Flüge wie von einem menschlichen Piloten erwartet aus. Zu einer Kreuzung der inneren Flüge (wie in Fall (a) zu sehen) kommt es in diesem Fall nicht. Durch die oben beschriebene Differenz zwischen der Angabe der Anfangsdistanz und der tatsächlichen Distanz zu Beginn des Fluges, der Geradeausflug lediglich bis zu einer Anfangsdistanz von $d_0 = 3500\,\mathrm{m}$ zu sehen. Ab einer Anfangsdistanz von $d_0 = 3250\,\mathrm{m}$ beträgt die tatsächliche Distanz zum linken Hindernis lediglich $d = d_0 - 500\,\mathrm{m} = 2750\,\mathrm{m}$. Dadurch wird bereits zu Beginn des Fluges ein *Corrective Alert* von DAIDALUS ausgegeben und das Ausweichmanöver unmittelbar nach Simulationsbeginn eingeleitet.
Bei einer Anfangsdistanz von $d_0 = 1750\,\mathrm{m}$ kommt es zu den ersten Verletzungen des *Corrective Volumes* des linken Hindernisses. Abbildung 4.8 zeigt dieses Szenario.

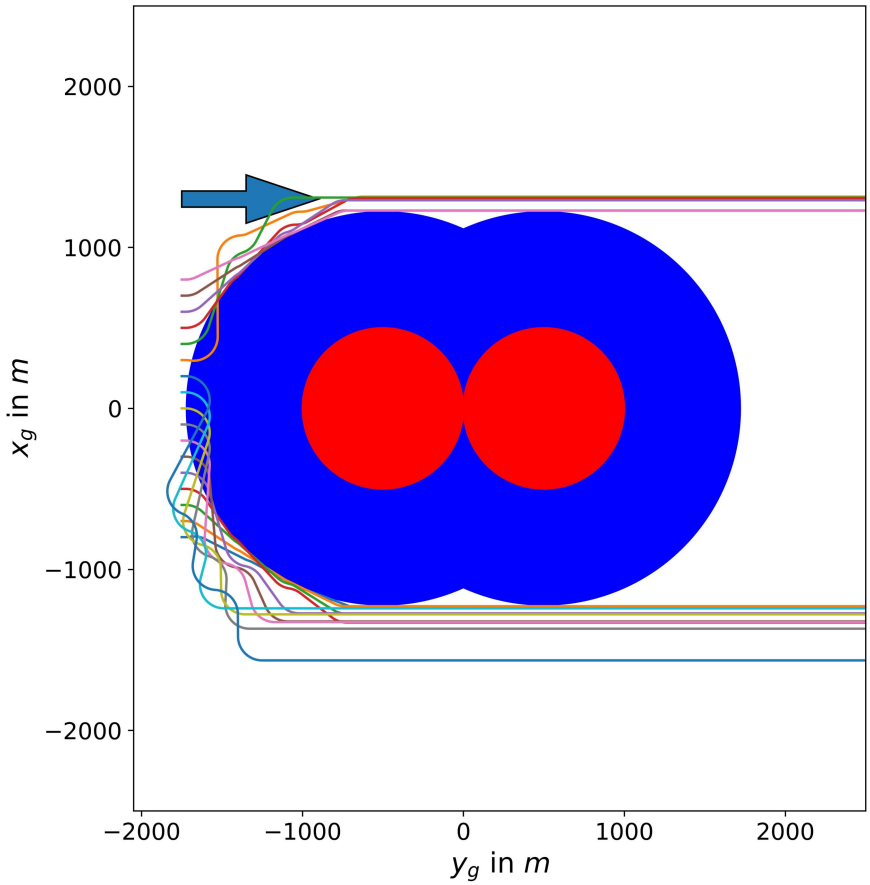


Abbildung 4.8 Anflug auf zwei statische Hindernisse mit Kurs $\chi = 90\,^\circ$ aus $d_0 = 1750\,\mathrm{m}$ Entfernung

Die oben beschriebene Differenz zwischen der angegebenen Anfangsdistanz d_0 und der tatsächlichen Anfangsdistanz muss auch in diesem Beispiel beachtet werden. Die tatsächliche Anfangsdistanz liegt bei $d = d_0 - 500\,\mathrm{m} = 1250\,\mathrm{m}$. Dadurch ergibt sich für den bei $x_g = 0\,\mathrm{m}$ startenden Flug eine Distanz bis zum Eintritt in das *Corrective Volume* von $d - DTHR = 1250\,\mathrm{m} - 1220\,\mathrm{m} = 30\,\mathrm{m}$. Bei einer Fluggeschwindigkeit von $v_A = 25\,\mathrm{m/s}$ bleibt dem Ownship eine Zeit von $1,2\,\mathrm{s}$ um den Eintritt in das *Corrective Volume* zu vermeiden. Wie in Fall (a) beschrieben ist auch in diesem Fall die Verletzung des *Corrective Volumes* aufgrund des minimalen Wenderadius des ALAADy-Demonstrators von $r_{w,min} = 136,6\,\mathrm{m}$ und der Aktualisierungsrate von DAIDALUS von $f = 1\,\mathrm{Hz}$ nicht zu vermeiden.
Fall (c) dieses Abschnittes wurde ebenfalls mit der dynamischen Simulation nachgebildet. Die statischen Hindernisse werden auch in der dynamischen Simulation erfolgreich umflogen. Zu einer Verletzung der *Separation Volumes* kommt es bei einer Anfangsdistanz von $d_0 \leq 2000\,\mathrm{m}$. Zu erklären ist die im Vergleich zu den Ergebnissen der kinematischen Simulation etwas höhere Anfangsdistanz wie in Fall (a) durch das Verhalten bei der Einleitung eines Kurvenfluges in der dynamischen Simulation.

4.1.2. Sackgasseneffekt

Bei der Kombination beziehungsweise Überschneidung von zwei (oder mehr) statischen Hindernissen entsteht eine V-förmige Einbuchtung durch die zylindrischen Volumen. Falls das

Ownship dem zylindrischen Verlauf eines Volumens folgt, stößt es nach einer gewissen Zeit (am Schnittpunkt der beiden Volumen) auf das Volumen des nächsten Hindernisses. Dadurch entsteht eine Art Sackgasse, die für diesen Abschnitt namensgebend ist. Abbildung 4.9 zeigt eine Schematische Darstellung des Szenarios zur Untersuchung des Sackgasseneffekts.

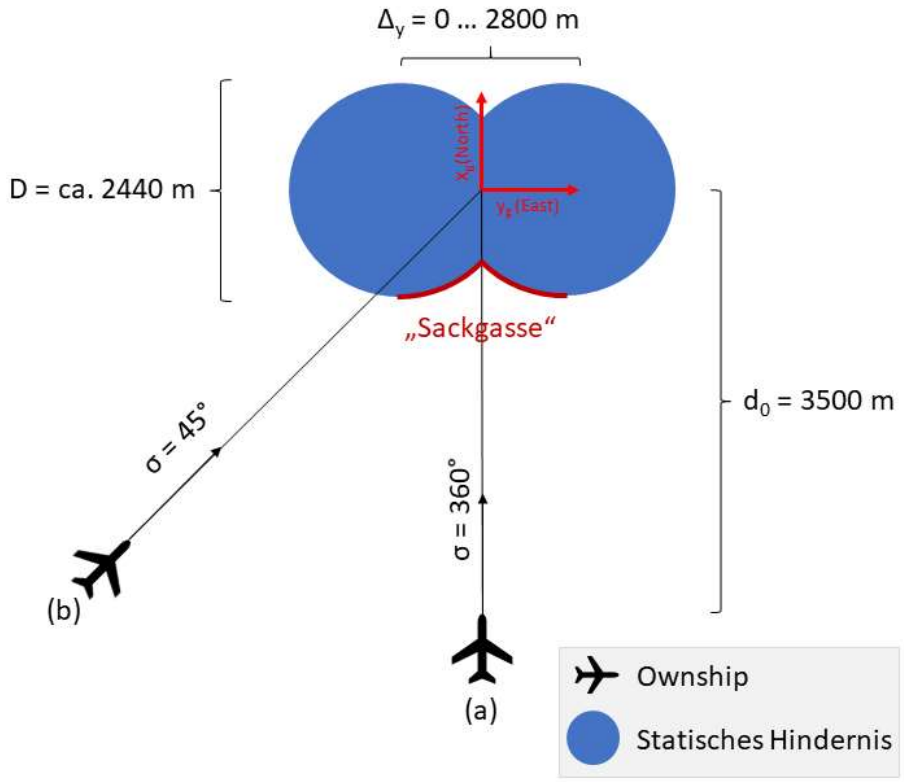


Abbildung 4.9 Schematische Darstellung Szenario Sackgasseneffekt

Wie in Abschnitt 4.1.1 werden zwei sich überschneidende Hindernisse platziert. Dabei wird die Entfernung der Mittelpunkte für die Verwendung des Standard-Alerters in 100 m-Schritten von $\Delta y = 0\,\mathrm{m}$ auf $\Delta y = 2800\,\mathrm{m}$. Das Ownship startet in einer Distanz von $d_0 = 3500\,\mathrm{m}$. Dabei werden zwei Fälle betrachtet:

- Fall (a): Anflug mit Kurs $\chi = 360\,^\circ$
- Fall (b): Anflug mit Kurs $\chi = 45\,^\circ$

Fall (a): Anflug mit Kurs $\chi = 360\,^\circ$

In Fall (a) kommt es in keinem Szenario zur Verletzung eines Volumens der statischen Hindernisse. Das Manöver führt jeweils rechts an den statischen Hindernissen vorbei. Exemplarisch für diese Szenarien zeigt Abbildung 4.10 den Flug bei einer Entfernung der Mittelpunkte der Hindernisse von $\Delta y = 1800\,\mathrm{m}$.

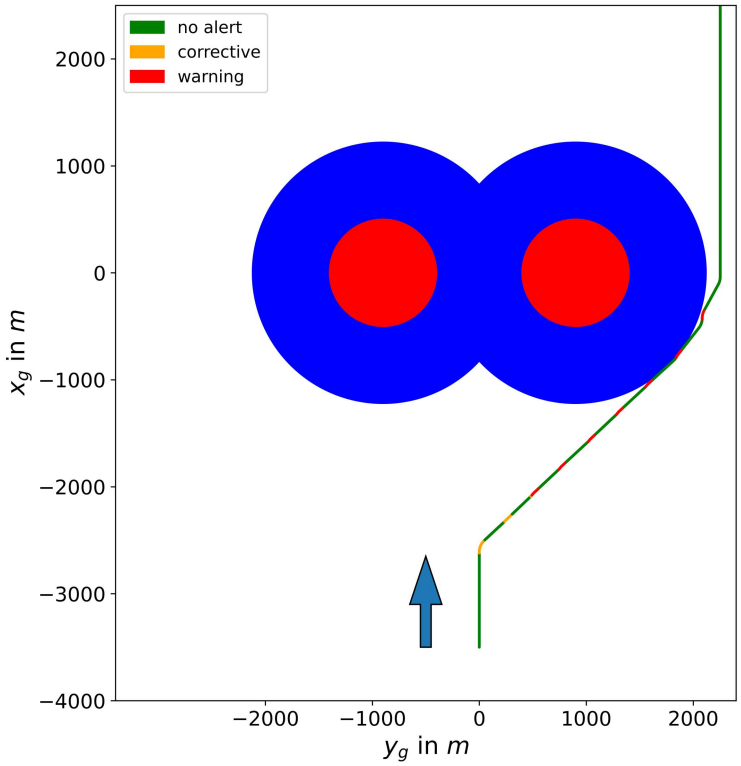


Abbildung 4.10 Sackgasseneffekt bei einer Hindernisentfernung von $\Delta\, y = 1800\,\mathrm{m}$ und einem Kurs von $\chi = 360\,^\circ$

Der dargestellte Flugpfad beginnt im Punkt $(x_g = -3500\,\mathrm{m}; y_g = 0\,\mathrm{m},)$ mit Kurs $\chi = 360\,^\circ$. Der blaue Pfeil veranschaulicht den Kurs zu Beginn des Fluges. In der Legende sind die *Alert Level* (*no alert*, *corrective* und *warning*) mit ihrer farblichen Codierung zum Flugpfad angegeben. Ein *Preventive Alert* ist nicht dargestellt, da die Parameter des *Preventive Volumes* denen des *Corrective Volumes* entsprechen. Der Flugpfad ist in regelmäßigen Abständen orange oder rot eingefärbt. Die dort vorherrschenden Warnungen entstehen durch ein kurzes Zurückschwenken auf den ursprünglichen Kurs von $\chi = 360\,^\circ$ nachdem für `t_sec` Sekunden keine Warnung besteht (vgl. Fall (a) in Abschnitt 4.1.1 und Codebeispiel in Abschnitt 3.1.2).
Je weiter die zwei statischen Hindernisse von einander entfernt sind, desto später wird das Ausweichmanöver gestartet. Durch einen wachsenden Abstand $\Delta\, y$ vertieft sich die zwischen den Hindernissen entstehende Sackgasse. Da das Ownship auf der x_g-Achse mit einem Kurs von $\chi = 360\,^\circ$ direkt auf den Schnittpunkt der Volumen zufliegt, vergrößert sich auch die Distanz zu diesem Schnittpunkt mit wachsendem $\Delta\, y$. Dadurch wird die Zeit, bis zum Erreichen von $TTHR$ und damit bis zur Auslösung des *Corrective Alerts* erhöht. Daraus resultiert ein längerer Geradeausflug entlang der x_g-Achse.
Für den oben dargestellten Flug wird das Ausweichmanöver bei einer Entfernung zum Ursprung von $d = 2620\,\mathrm{m}$ gestartet. An dieser Stelle wird von DAIDALUS erstmals für diesen Flug ein *Corrective Alert* gemeldet.
Das Ausweichmanöver ähnelt sich für alle Flüge bis zu einer Distanz der beiden Mittelpunk-

te der statischen Hindernisse von $\Delta y = 2400\,\mathrm{m}$. Bei diesem Flug überlappen die beiden Volumen sich mit einer Breite von knapp $20\,\mathrm{m}$. Zu einer Verletzung der *Separation Volumes* kommt es dabei nicht.
Bereits ab einer Entfernung der beiden Mittelpunkte der statischen Hindernisse von $\Delta y = 2500\,\mathrm{m}$ wird der komplette Flug mit dem Kurs $\chi = 360\,^\circ$ durchgeführt und über den gesamten Flug keine Warnung von DAIDALUS ausgegeben. Das Ownship durchfliegt damit eine Lücke mit einer Breite von knapp $60\,\mathrm{m}$. Da DAIDALUS die Abmessungen des Ownships nicht beachtet, deckt sich dieses Ergebnis mit dem vorab erwarteten Verhalten.

Fall (b): Anflug mit Kurs $\chi = 45\,^\circ$
Auch in Fall (b) - bei einem Anflug mit Kurs $\chi = 45\,^\circ$ - weichen alle Flüge zuverlässig den statischen Hindernissen aus, ohne deren *Separation Volumes* zu verletzen. Wie in Fall (a) werden auch in Fall (b) die Hindernisse über ein rechts vorbeiführendes Ausweichmanöver umflogen. Die *Separation Volumes* der statischen Hindernisse werden dabei in keinem Flug verletzt. Abbildung 4.11 zeigt exemplarisch einen Flug für Fall (b). Die Mittelpunkte der zwei statischen Hindernisse sind dabei $\Delta y = 1800\,\mathrm{m}$ voneinander entfernt.

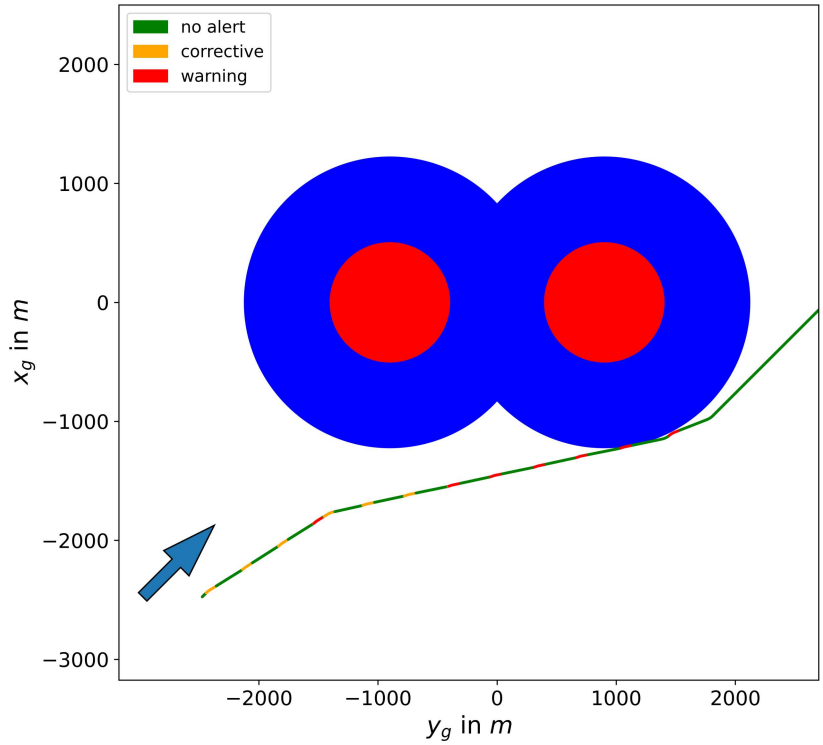


Abbildung 4.11 Sackgasseneffekt bei einer Hindernisentfernung von $\Delta y = 1800\,\mathrm{m}$ und einem Kurs von $\chi = 45\,^\circ$

Im Gegensatz zu Fall (a) wird bei einem Anflug mit einem Kurs von $\chi = 45\,^\circ$ nicht zwischen beiden Hindernissen durchgeflogen, sofern sie weit genug voneinander entfernt sind. Abbil-

dung 4.12 zeigt den Flug mit Kurs $\chi = 45\,^\circ$ und einer Distanz der Mittelpunkte der zwei statischen Hindernisse von 2800 m. Dabei ist mit einer gestrichelten Linie der ursprünglich geplante Kurs von $\chi = 45\,^\circ$ eingezeichnet.

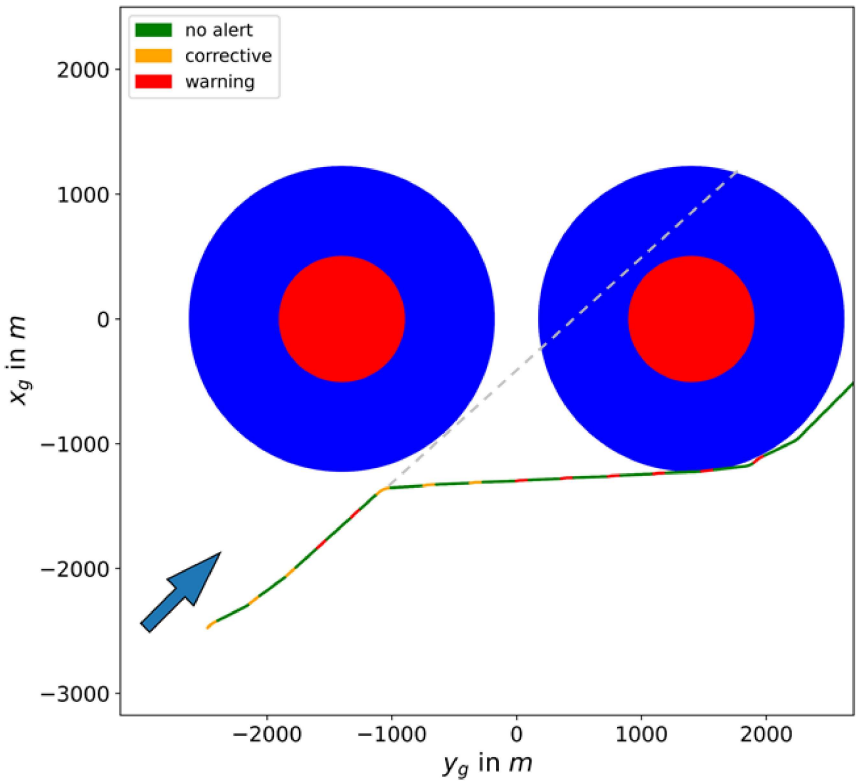


Abbildung 4.12 Sackgasseneffekt bei einer Hindernisentfernung von $\Delta y = 2800\,\mathrm{m}$ und einem Kurs von $\chi = 45\,^\circ$

Wie in Abbildung 4.12 zu erkennen ist, würde ein Ausbleiben des Ausweichmanövers dazu führen, dass das Ownship das Volumen des östlichen Hindernisses verletzen würde. Dass DAIDALUS stattdessen einen Ausweichkurs von etwa $\chi = 85\,^\circ$ vorschlägt ist damit zu begründen, dass DAIDALUS bei der Berechnung des Ausweichmanövers einen aktuellen Zustand betrachtet und die Geschwindigkeitsvektoren des Ownships und der Hindernisse als konstant annimmt. Die Geschwindigkeitsvektoren der statischen Hindernisse sind für alle Dimensionen mit $0\,\mathrm{m/s}$ definiert. Da DAIDALUS lediglich geradlinige Ausweichempfehlungen berechnen kann, wird stattdessen Kursänderung ausgegeben, bei der die kürzeste Zeitdauer im Volumen der statischen Hindernisse verbracht wird. Bei einem Kurs von etwa $\chi = 85\,^\circ$ fällt diese Zeitdauer auf $0\,\mathrm{s}$. Da diese die nächstgelegene Möglichkeit mit einer Aufenthaltsdauer von $0\,\mathrm{s}$ in den Volumen der statischen Hindernisse ist, wird diese als Ausweichkurs vorgegeben. Eine zweite Möglichkeit, keinen Moment in den *Separation Volumes* zu verbringen, wäre zum Beispiel ein Kurs von etwa $\chi = 280\,^\circ$. Dies erfordert allerdings eine deutlich größere Kurskorrektur und ist damit nicht die nächstgelegene Möglichkeit. Weiterführende Tests zeigten, dass ab einer Distanz der beiden Mittelpunkte der statischen Hindernisse von $\Delta y = 3100\,\mathrm{m}$ zwischen den beiden Volumen durchgeflogen wird.

4.1.3. Geofence

Wie in Abbildung 3.3 gezeigt, können Geofences mit der Aneinanderreihung mit Überschneidung einzelner statischer Hindernisse nachgebildet werden. Ein *keep-out Geofence* entspricht in dieser Arbeit einer Aneinanderreihung von mehreren statischen Hindernissen, denen einmalig ausgewichen werden muss. Dabei besteht dieses einmalige Ausweichmanöver gegebenenfalls aus mehreren Kursänderungen (die in jedem Zyklus von DAIDALUS aktualisiert werden). Diese Problemstellung ist vergleichbar mit den zuvor präsentierten Szenarien mit statischen Hindernissen. Dagegen ist das Ownship innerhalb eines *keep-in Geofences* von statischen Hindernissen umzingelt und muss diesen permanent ausweichen. Um das Verhalten innerhalb eines keep-in Geofences zu beurteilen, wird ein quadratischer Geofence mit einer Seitenlänge von 10 km erzeugt. Durch die Ausdehnung der Volumen der statischen Hindernisse ergibt dies einen Bereich von ca. 8 km x 8 km in dem sich das Ownship problemlos bewegen kann. Abbildung 4.13 zeigt den schematischen Aufbau dieses Szenarios.

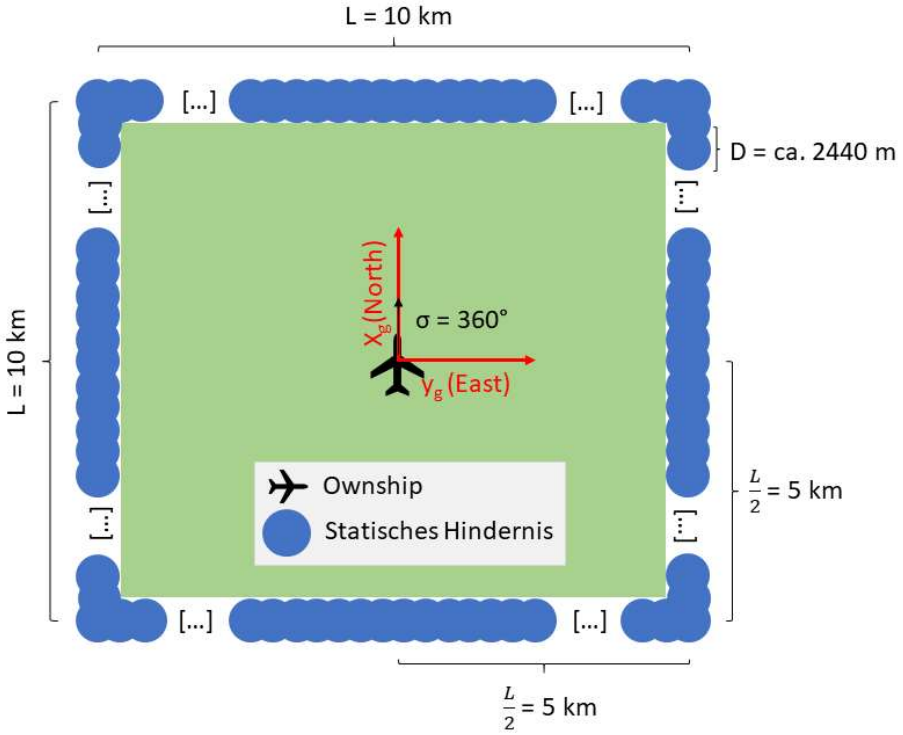
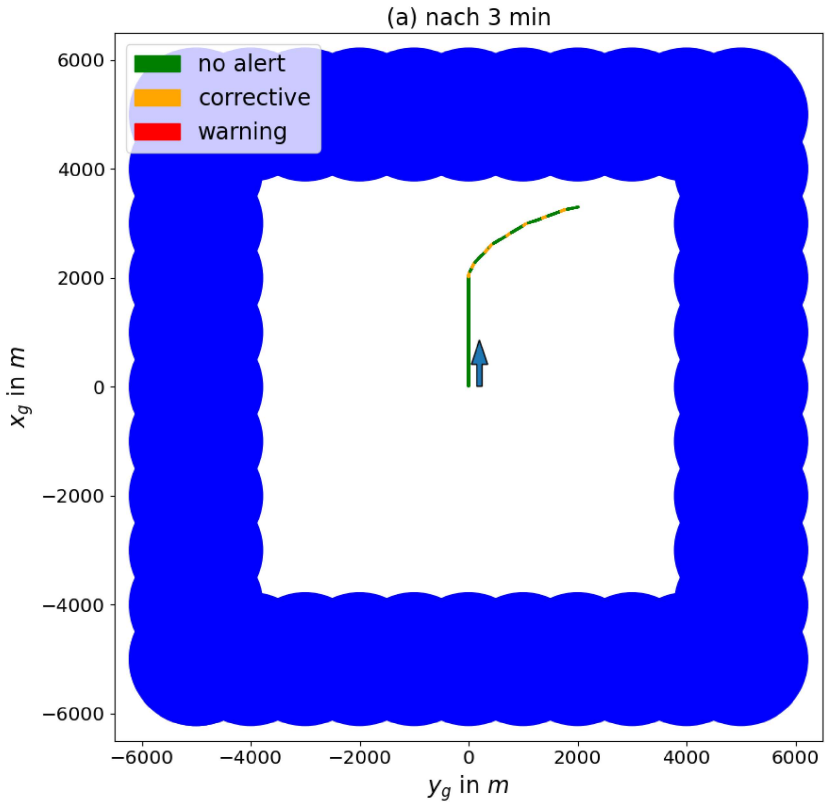


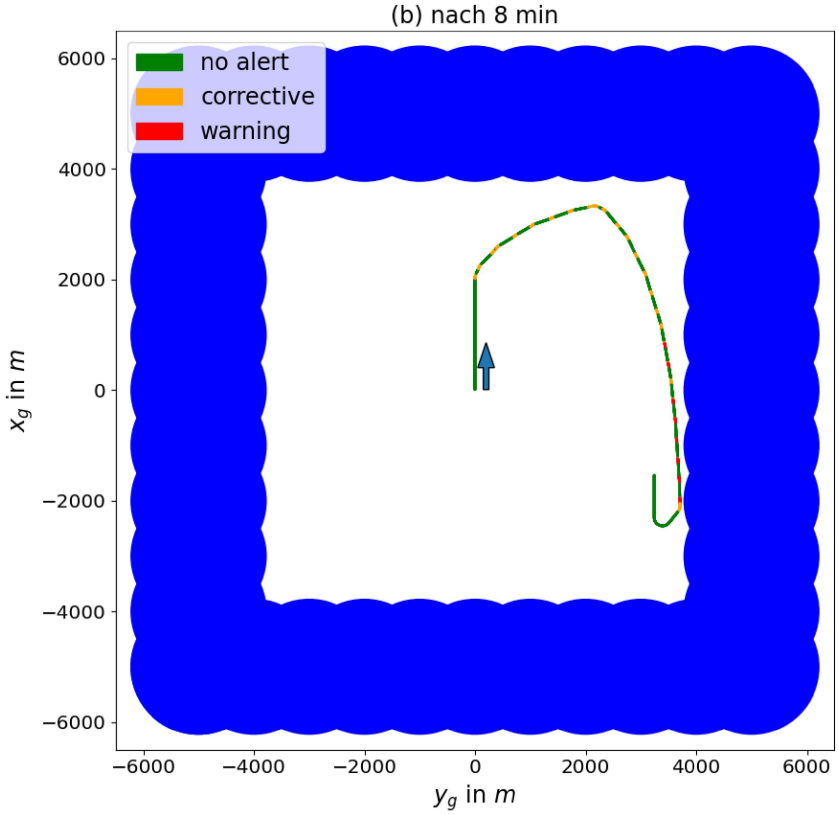
Abbildung 4.13 Schematische Darstellung Szenario Geofence

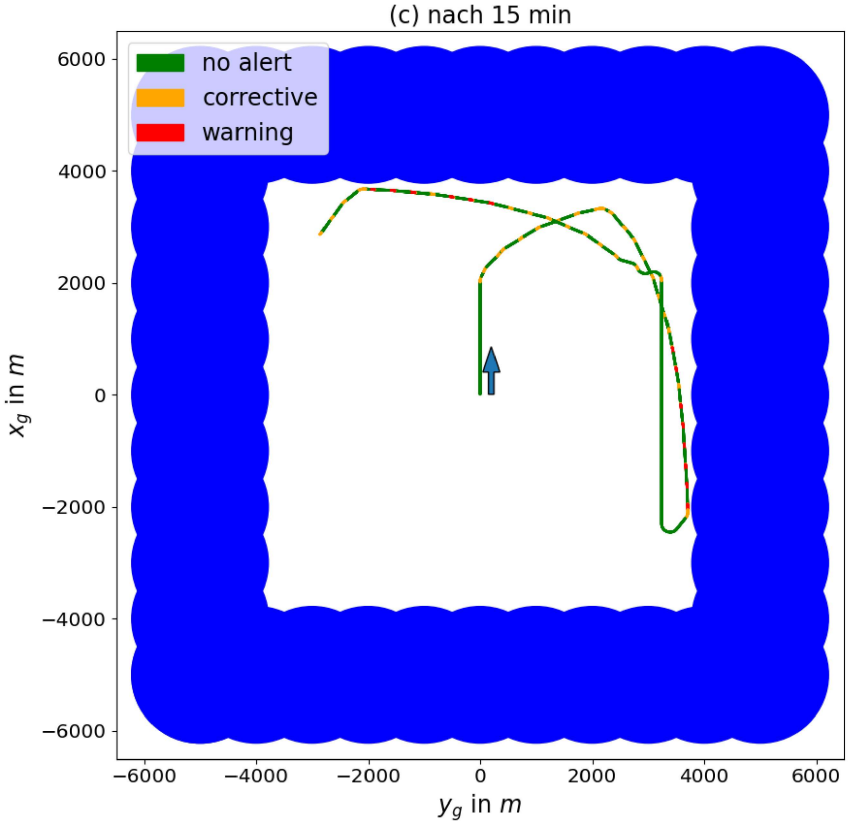
Die einzelnen statischen Hindernisse werden so aneinandergereiht, dass sie sich jeweils mindestens zu 50 % überlappen. Damit wird eine möglichst glatte Außenkontur des in grün dargestellten Geofences erreicht. Das Ownship wird im Zentrum des Geofences mit Kurs $\chi = 360\,^\circ$ gestartet und das Verhalten nach $t_{end} = 5 : 10\,\mathrm{h}$ [2] bewertet. Das Ownship fliegt mit einer konstanten Fluggeschwindigkeit von $v_A = 25\,\mathrm{m/s}$ [3] und legt dabei eine Strecke von $s = 1080\,\mathrm{km}$ zurück. Der Dauertest soll zeigen, wie zuverlässig DAIDALUS funktioniert, wenn das Ownship umgeben von statischen Hindernissen ist und kein Ausweg zwischen den Hindernissen zu finden ist. Um den Flugpfad nachvollziehen zu können bietet Abbildung 4.14

[2] $18.600\,\mathrm{s}$
[3] $90\,\mathrm{km/h}$

vier Teilplots, die alle denselben Flug darstellen. Dabei ist in Teilplot (a) der Flugpfad nach $t_1 = 3\,\mathrm{min}$, in Teilplot (b) ist der Flugpfad nach $t_2 = 8\,\mathrm{min}$ zu sehen. Teilplot (c) zeigt den Flugpfad nach $t_3 = 15\,\mathrm{min}$, Teilplot (d) zeigt den Flugpfad nach $t_4 = 30\,\mathrm{min}$. Durch den blauen Pfeil nahe des Ursprungs ist jeweils der initiale Kurs zum Startzeitpunkt $t_0 = 0\,\mathrm{s}$, $\chi = 360\,^\circ$, angedeutet.




(b) nach 8 min
no alert
corrective
warning
6000
4000
2000
0
−2000
−4000
−6000
x_g in m
−6000
−4000
−2000
0
2000
4000
6000
y_g in m


(c) nach 15 min
no alert
corrective
warning
6000
4000
2000
0
−2000
−4000
−6000
x_g in m
−6000
−4000
−2000
0
2000
4000
6000
y_g in m

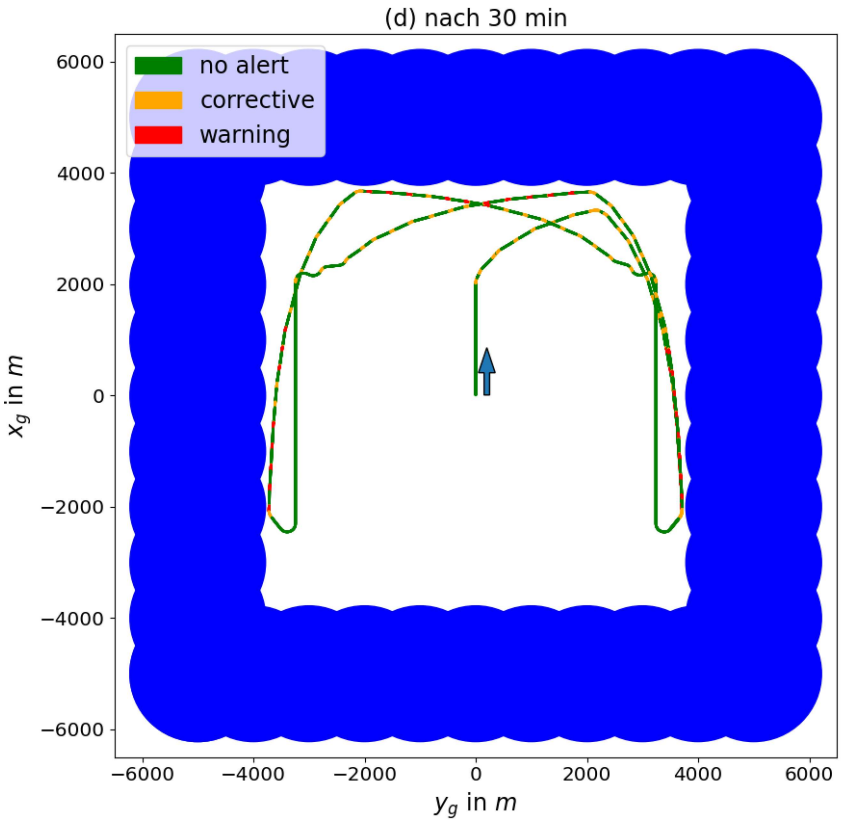


Abbildung 4.14 Collage Szenario keep-in Geofence

Die Einfärbung des Flugpfades zeigt jeweils, an welchen Orten des Flugpfades eine Warnung von DAIDALUS ausgegeben wurde. Das *Alert Level* der jeweiligen Farbe ist der Legende oben links in den Plots zu entnehmen.
Wie in Teilplot (a) zu erkennen ist, beginnt DAIDALUS bei $x_g \approx 2000\,\mathrm{m}$ mit dem ersten Ausweichmanöver in positive x_g-/y_g-Richtung. Durch die Lage der statischen Hindernisse bei $x_g = 5000\,\mathrm{m}$ ergibt sich eine Differenz von $\Delta x_g = 3000\,\mathrm{m}$. Das Verhalten deckt sich mit dem in Abschnitt 4.1.1 erkannten Verhalten. Anschließend nähert sich das Ownship den Grenzen der statischen Hindernisse immer näher an. Dabei werden mehrere Kurskorrekturen (leichte Rechtskurven) durchgeführt. Damit wird das Einfliegen in eines der *Separation Volumes* verhindert bzw. verzögert. Bei einer Position von ca. ($x_g = 3300\,\mathrm{m}$; $y_g = 2100\,\mathrm{m}$) wird eine enge Rechtskurve von knapp $60\,^\circ$ durchgeführt. Hier hat sich erneut ein Abstand von ca. $\Delta y_g = 3000\,\mathrm{m}$ zu den bei $y_g = 5000\,\mathrm{m}$ liegenden statischen Hindernissen eingestellt.
Im weiteren Verlauf (Teilplot (b)) fliegt das Ownship entlang der statischen Hindernisse in negative x_g-Richtung, wobei sich das Ownship in y_g-Richtung immer weiter an den Geofence annähert. Am Ende des Flugpfades von Teilplot (b) ist eine enge Rechtskurve zu erkennen. Nach Beenden dieser Rechtskurve folgt ein Stück von etwa $5000\,\mathrm{m}$, in denen das Ownship parallel zur x_g-Achse fliegt. Dieser Abschnitt ist auf die Codezeilen 45 bis 49 im Codebeispiel in Abschnitt 3.1.2 zurückzuführen. Dabei handelt es sich um die Rückführung auf den ursprünglich geflogenen Kurs von $\chi = 360\,^\circ$. Nachdem bei $x_g \approx 2000\,\mathrm{m}$ erneut einen *Corrective Alert* von DAIDALUS gemeldet wird, wird ein Ausweichmanöver in negative x_g-Richtung durch eine Kurve mit knapp $100\,^\circ$ eingeleitet. Das Ausweichmanöver sorgt dafür, dass DAIDA-

LUS für $10\,\mathrm{s}$ keine Warnung ausgibt und das Ownship somit erneut auf den ursprünglichen Kurs von $\chi = 360\,^\circ$ dreht (vgl. oben und Codezeilen 45-49 in Abschnitt 3.1.2). Die Kurve zurück auf den ursprünglichen Kurs wird allerdings noch vor Beendung der Kurve unterbrochen, da DAIDALUS erneut einen *Corrective Alert* ausgibt. Im Folgenden nähert sich das Ownship den Volumen der statischen Hindernisse immer näher an. Bei einer Position von etwa ($x_g = 3600\,\mathrm{m}$; $y_g = -3000\,\mathrm{m}$) ist die geringste Distanz zu den statischen Hindernissen erreicht und DAIDALUS empfiehlt ein Ausweichmanöver in negative x_g-/y_g-Richtung, das das Ownship folglich fliegt. Ab dieser Stelle ergibt sich für die linke Seite (negative y_g-Koordinaten) ein spiegelgleicher Flugpfad zur rechten Seite (positive y_g-Koordinaten). Zu erkennen ist dies in Teilplot (d). Nachfolgende Abbildung 4.15 zeigt den gesamten Flug mit einer Dauer von $\Delta t = 5:10\,\mathrm{h}$.

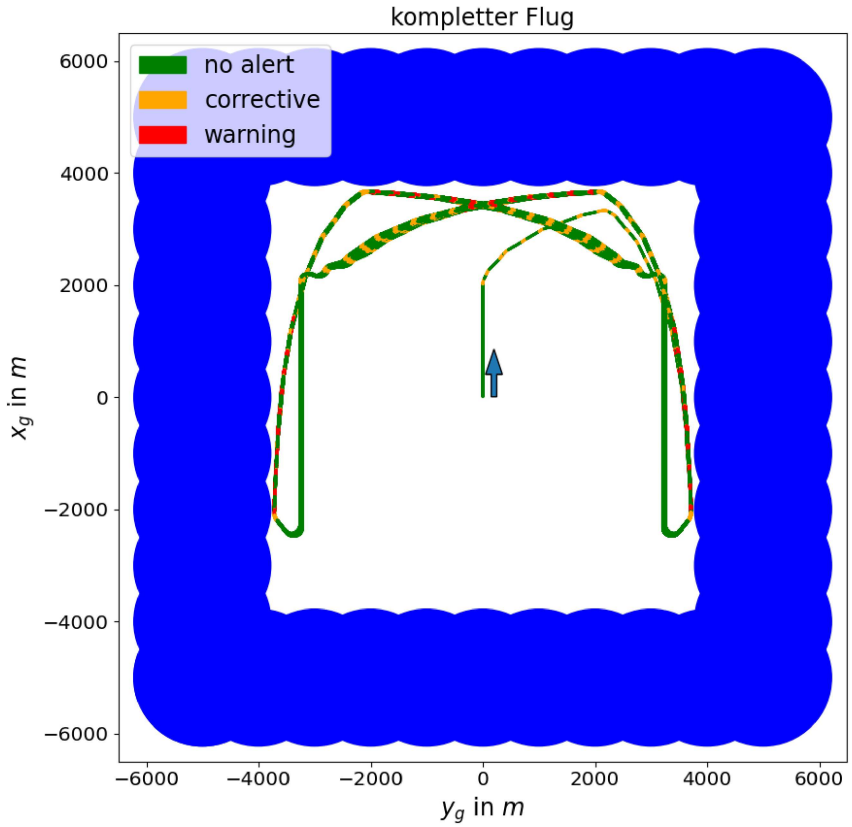


Abbildung 4.15 Gesamter Flug Szenario Geofence

Es ist zu erkennen, dass die Flugpfade der einzelnen Runden nicht genau aufeinander liegen sondern leicht voneinander abweichen. Das kann durch die Aktualisierungsrate des DAIDALUS Algorithmus erklärt werden. Die Aktualisierungsrate wird im Rahmen dieser Arbeit mit $f = 1\,\mathrm{Hz}$ aktualisiert. Durch eine geringfügig andere Position oder Orientierung des Ownships zum jeweiligen Aktualisierungszeitpunkt kommt es im Vergleich zur jeweiligen Berechnung in der vorherigen Runde zu leichten Unterschieden in den Ausgangswerten der Berechnung. Dadurch ergeben sich auch leicht abweichende Ergebnisse im Vergleich zu den Berechnungen der vorherigen Runde.

Der gesamte Flug von $t_{Flugdauer} = 5:10\,\mathrm{h} = 310\,\mathrm{min}$ Dauer wurde mit der in Tabelle 4.1 aufgeführten Hardware in einer Zeit von etwa $\Delta t = 2:30\,\mathrm{h} = 150\,\mathrm{min}$ simuliert. Im

Durchschnitt wurde eine Minute Flugzeit damit in

$$\frac{150\,\text{min}}{310\,\text{min}} = 0,48\,{}^{\text{min}}/_{\text{min}} = 29,03\,{}^{\text{s}}/_{\text{min}}$$

berechnet.
Das Szenario eines Geofences wurde außerdem mit der dynamischen Simulation durchgeführt. Bis auf leichte Unterschiede in den Flugpfaden sind die Ergebnisse identisch. Die unterschiedlichen Flugpfade sind dabei Folge der unterschiedlichen Bewegungsmodelle beider Simulationen.

4.1.4. Flugkorridor

Im durchgeführten Test wird ein vorgegebener Flugkorridor durch zwei ineinander liegende Quadrate definiert. In Abbildung 4.16 ist dieser Flugkorridor grün eingefärbt.

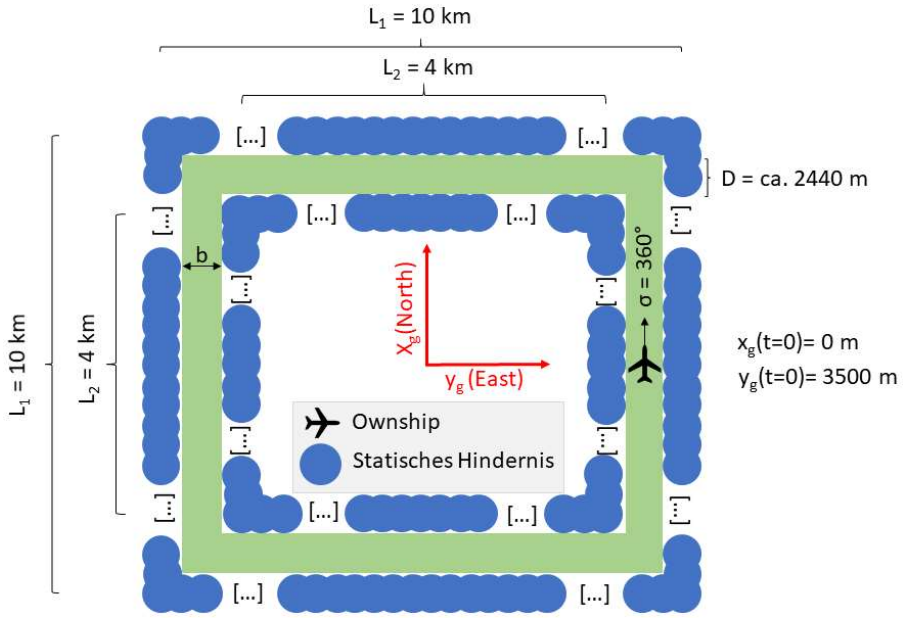


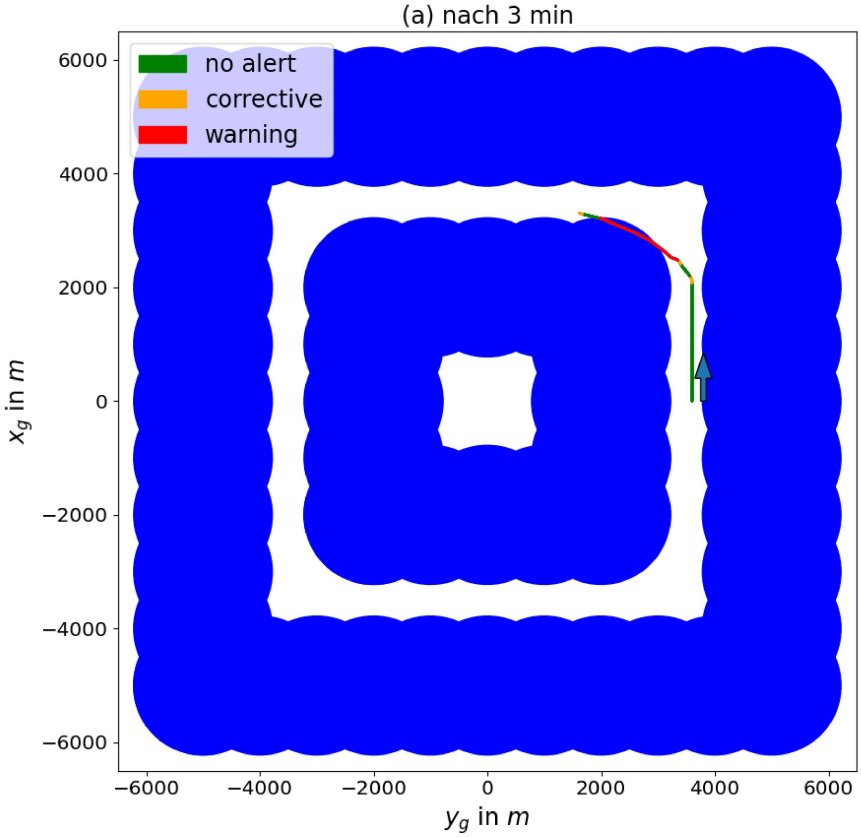
Abbildung 4.16 Schematische Darstellung Szenario Flugkorridor

Die Quadrate, die den Flugkorridor definieren, haben eine Seitenlänge von $L_1 = 10\,\text{km}$ bzw. $L_2 = 4\,\text{km}$. Die Quadrate liegen symmetrisch zum Koordinatenursprung. Durch die beiden Quadrate und den Radius der Volumen der statischen Hindernisse ergibt sich ein Flugkorridor mit einer Breite von:

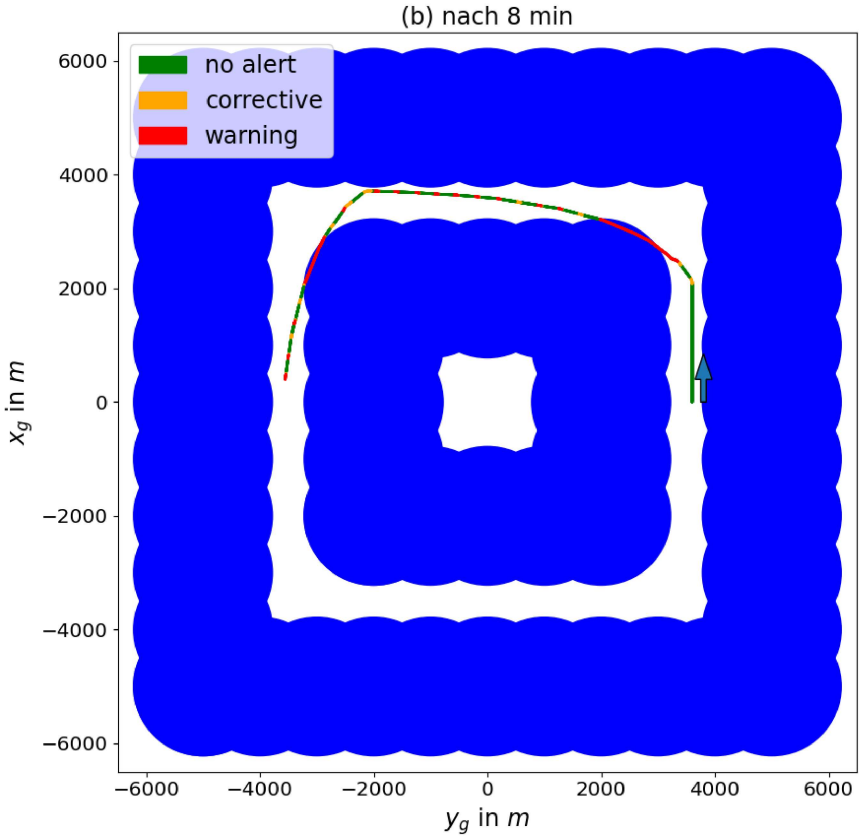
$$B = \frac{10.000\,\text{m} - 4.000\,\text{m}}{2} - 2 \cdot 1.220\,\text{m} = 560\,\text{m}$$

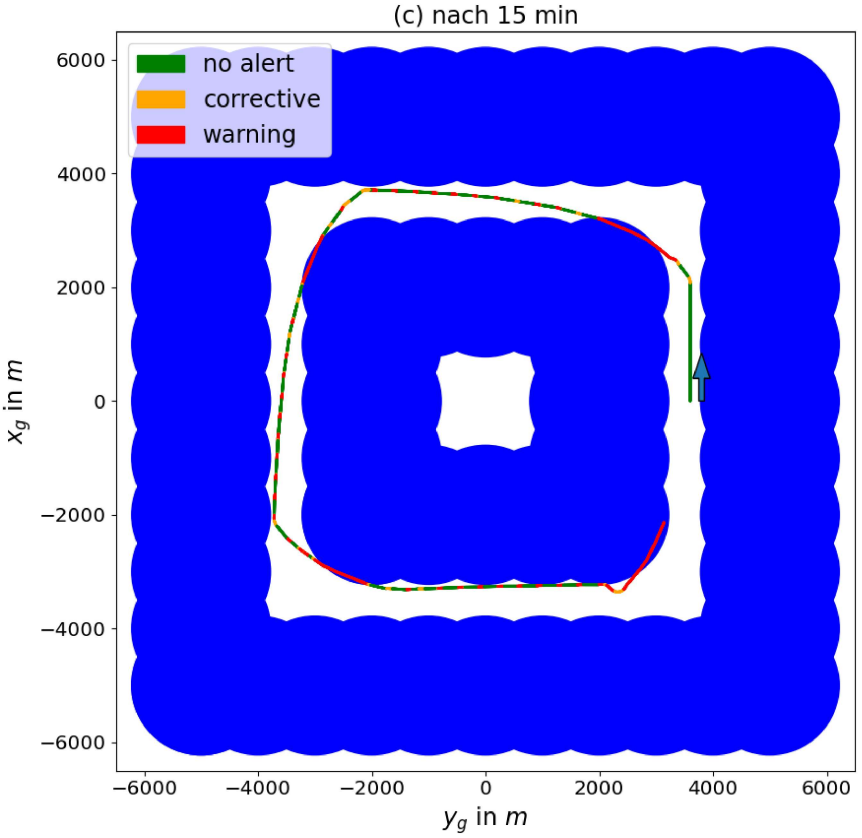
Die Startposition des Ownships liegt bei ($x_g = 0\,\text{m}$; $y_g = 3500\,\text{m}$). Das Ownship startet mit einem Kurs $\chi = 360\,°$ und einer Fluggeschwindigkeit von $v_A = 25\,{}^{\text{m}}/_{\text{s}}$.
Um den Flugpfad nachvollziehen zu können bietet Abbildung 4.17 eine Collage eines einzigen Fluges. Dabei ist in Teilplot (a) der Flugpfad nach $t_1 = 3\,\text{min}$, in Teilplot (b) ist der

Flugpfad nach $t_2 = 8\,\mathrm{min}$ zu sehen. Teilplot (c) zeigt den Flugpfad nach $t_3 = 15\,\mathrm{min}$, in Teilplot (d) ist der Flugpfad nach $t_4 = 30\,\mathrm{min}$ abgebildet. Durch den blauen Pfeil nahe des Startpunkts ist jeweils der initiale Kurs zum Startzeitpunkt $t_0 = 0\,\mathrm{s}, \chi = 360\,^\circ$ angedeutet. Die von DAIDALUS ausgegebene Warnung entspricht der in der Legende angegebenen Farbcodierung: Liegt keine Meldung vor, so ist der Flugpfad grün eingefärbt. Bei einem *Corrective Alert* ist der Flugpfad orange und bei einem *warning alert* ist der Flugpfad rot gefärbt.



In Teilplot (a) von Abbildung 4.17 ist das erste von DAIDALUS ausgegebene Ausweichmanöver zu erkennen. Bei Erreichen der Position ($x_g = 2100\,\mathrm{m}; y_g = 3500\,\mathrm{m}$) meldet DAIDALUS einen *Corrective Alert* und empfiehlt ein Ausweichmanöver nach links. Im Laufe des Ausweichmanövers wird die Kurve enger, bis der Flugpfad schließlich das *Corrective Volume* des statischen Hindernisses bei ($x_g = 2500\,\mathrm{m}; y_g = 2500\,\mathrm{m}$) schneidet. Dabei nähert sich das Ownship dem Hindernis bis auf eine Distanz von $d_1 = 1107,5\,\mathrm{m}$ an. Bei genauerer Betrachtung der Logdaten fällt auf, dass das Ausweichmanöver ursprünglich einen Kurs von $\chi = 167\,^\circ$ vorsieht. Kurz nach der Einleitung dieses Ausweichmanövers wird die Kurskorrektur allerdings moderater gestaltet und ein Kurs von $\chi = 322\,^\circ$ wird von DAIDALUS ausgegeben. Vorerst ungeklärt bleibt, warum DAIDALUS ein Ausweichmanöver empfiehlt, das as *Corrective Volume* des Hindernisses schneidet. Würde das Ownship den originalen Kurs von $\chi = 360\,^\circ$ noch etwa $1\,\mathrm{km}$ länger halten, so könnte es fehlerfrei um die erste Ecke des Flugkorridors fliegen.


(b) nach 8 min
no alert
corrective
warning
6000
4000
2000
0
−2000
−4000
−6000
x_g in m
−6000
−4000
−2000
0
2000
4000
6000
y_g in m


(c) nach 15 min
no alert
corrective
warning
6000
4000
2000
0
−2000
−4000
−6000
x_g in m
−6000
−4000
−2000
0
2000
4000
6000
y_g in m

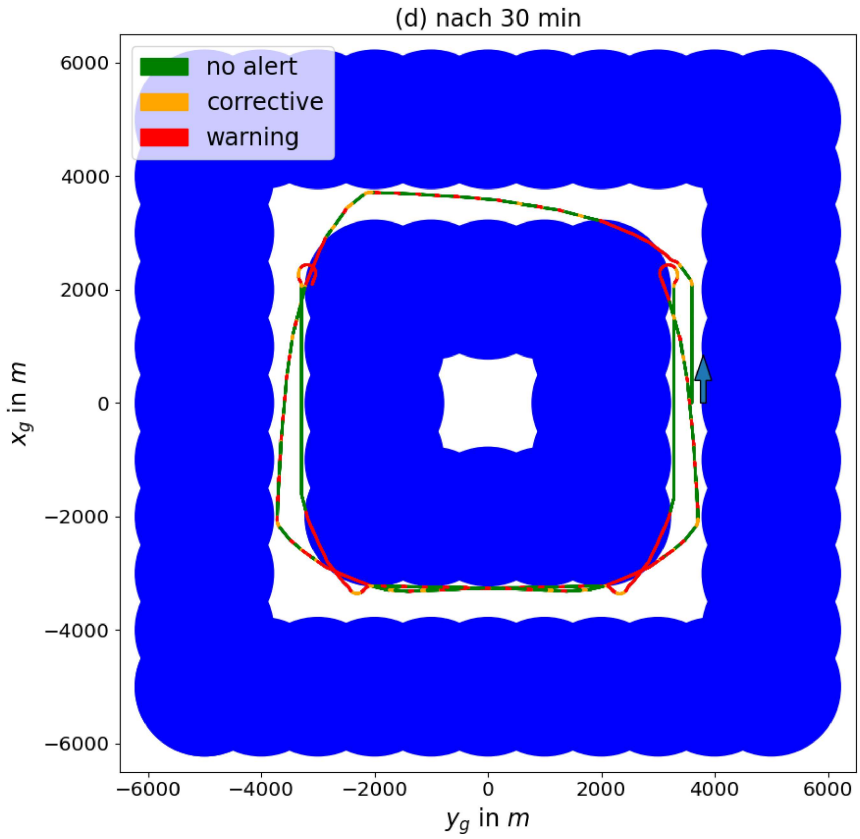


Abbildung 4.17 Collage Szenario Flugkorridor

Nachdem das *Corrective Volume* des statischen Hindernisses bei ($x_g = 2500\,\mathrm{m}$; $y_g = 2500\,\mathrm{m}$) verlassen ist, nähert sich das Ownship dem äußeren Rand (größere x_g-Koordinate) des Flugkorridors. Bevor es zu einer Verletzung der außen gelegenen Volumen kommt - bei etwa ($x_g = 4000\,\mathrm{m}$; $y_g = -3000\,\mathrm{m}$) - wird von DAIDALUS während eines *Corrective Alerts* erneut ein Ausweichmanöver nach links (Kurs etwa $\chi = 200\,°$) empfohlen. Bei diesem Ausweichmanöver kommt es erneut zur Verletzung eines *Corrective Volumes*. Das Ownship nähert sich auf eine Distanz von $d_2 = 1154\,\mathrm{m}$ an das statische Hindernis bei ($x_g = 2500\,\mathrm{m}$; $y_g = -2500\,\mathrm{m}$) an. Dieses Verhalten wiederholt sich, wie in Teilplot (c) zu sehen, an der nächsten Ecke ($x_g = -2500\,\mathrm{m}$; $y_g = -2500\,\mathrm{m}$). Die Distanz zu diesem Hindernis beträgt am nächsten Punkt $d_3 = 1155\,\mathrm{m}$.

An der Ecke des statischen Hindernisses bei ($x_g = -2500\,\mathrm{m}$; $y_g = 2500\,\mathrm{m}$) berechnet DAIDALUS nach einem *Corrective Alert* anstatt eines Ausweichmanövers, das wie oben das *Corrective Volume* des Hindernisses schneidet, ein Ausweichmanöver mit einem Kurs des Ownships von etwa $\chi = 150\,°$. Die Kurskorrektur ist in Teilplot (d) zu sehen. Dabei wird das Ausweichmanöver direkt auf die Rechtskurve folgend mit einer Linkskkurve von etwa $260\,°$ berechnet, was zu einem Kurs des Ownships von etwa $\chi = 250\,°$ führt. Bei der zuletzt beschriebenen Linkskurve schneidet das Ownship das *Corrective Volume* des statischen Hindernisses bei ($x_g = -2500\,\mathrm{m}$; $y_g = 2500\,\mathrm{m}$). Das Ownship nähert sich dem Hindernis auf bis zu $d_4 = 1085,5\,\mathrm{m}$.

Nachdem das Ownship wie oben beschrieben am rechten unteren Eck gewendet hat, fliegt es eine komplette Runde im Uhrzeigersinn. Hierbei wird jeweils das *Corrective Volume* der

statischen Hindernisse an den inneren Eckpunkten geschnitten. Die Distanzen zu den Mittelpunkten der Hindernisse bewegen sich im Bereich der oben genannten Werte (d_1 bis d_4). Zu sehen ist der komplette Flugpfad in Abbildung 4.18.

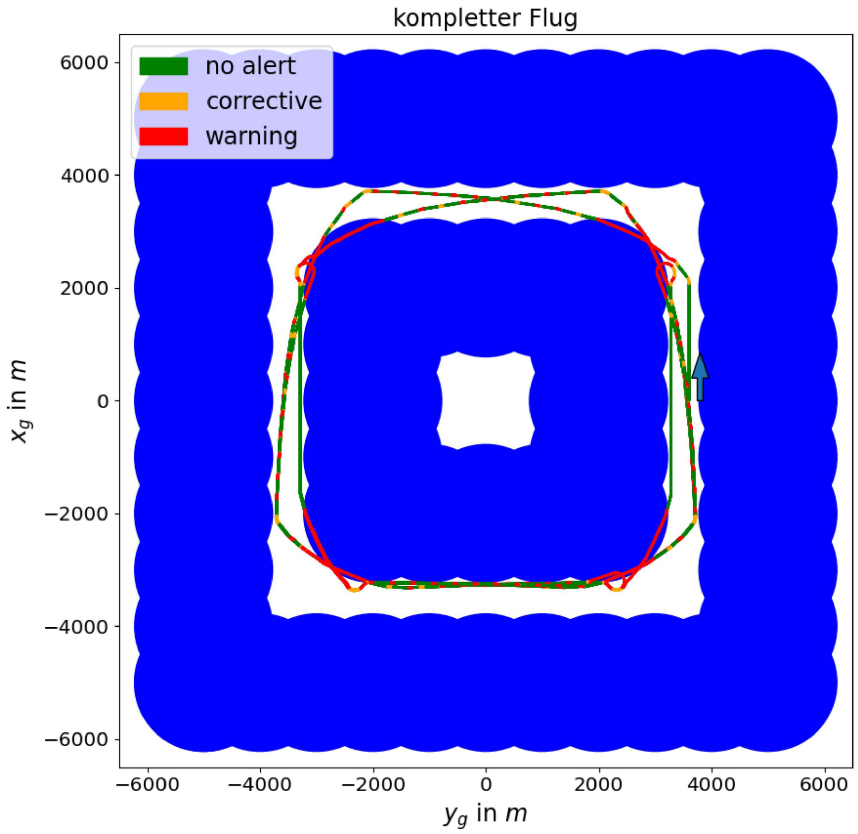


Abbildung 4.18 Gesamter Flug Szenario Flugkorridor

Nachdem das Ownship eine Runde im Uhrzeigersinn im Flugkorridor geflogen ist, wendet es erneut. Diesmal geschieht das Wendemanöver an der linken unteren Ecke ($x_g = -2500\,\mathrm{m}$; $y_g = -2500\,\mathrm{m}$). Das Manöver wird dabei spiegelgleich zum ersten Wendemanöver an der rechten unteren Ecke geflogen. Im Folgenden wird eine Runde entgegen des Uhrzeigersinns geflogen und wieder am rechten unteren Eck gewendet. Daraus ergibt sich beinahe eine Endlosschleife. Nach einigen Runden wird erneut an der rechten unteren Ecke gewendet. Dabei erfolgt das Wendemanöver bei etwa ($x_g = -2000\,\mathrm{m}$; $y_g = 3000\,\mathrm{m}$) und endet mit einem Kurs des Ownships von $\chi = 360\,°$. Im Folgenden wird bei ($x_g = 3800\,\mathrm{m}$; $y_g = 3500\,\mathrm{m}$) eine $180\,°$-Wende vollzogen. Bei diesem Wendemanöver fliegt das Ownship bis an die Volumen des statischen Hindernisses heran, verletzt diese allerdings nicht. Schließlich endet der Flug nach einer Simulationszeit von $t_{Flugdauer} = 3\,\mathrm{h}$. Die Berechnung dieses $3\,\mathrm{h}$-Fluges benötigt mit der verwendeten Hardware (vgl. Tabelle 4.1) innerhalb von $\Delta t = 29\,\mathrm{min}$. Damit ergibt sich eine durchschnittliche Berechnungszeit für eine Stunde Flugzeit von

$$\frac{\Delta t}{t_{Flugdauer}} = \frac{29\,\mathrm{min}}{3\,\mathrm{h}} = 9,6\,{}^{\mathrm{min}}/_{\mathrm{h}} = 580\,{}^{\mathrm{s}}/_{\mathrm{h}}$$

Die Berechnung des hier beschriebenen Flugkorridor-Szenarios läuft fast 3,8 mal langsamer ab als die Berechnung des in Abschnitt 4.1.3 beschriebenen keepin-Geofence-Szenarios. Zu erklären ist dies durch die zusätzlichen Hindernisse, die von DAIDALUS bei der Berechnung des Ausweichmanövers und der Warnungen beachtet werden müssen.
Das kurzzeitige Aufkommen einer Ausweichempfehlung mit Kurs $\chi = 167\,^\circ$ vor der ersten Kurve des Flugkorridors lässt vermuten, dass DAIDALUS den freien, zu diesem Zeitpunkt hinter dem Ownship liegenden, Flugkorridor als mögliche Ausweichrichtung erkennt. Der freie Bereich bietet dem Algorithmus die längste Flugzeit, ohne in ein *Separation Volume* eines Hindernisses einzufliegen. Damit verliert der vor dem Ownship liegende freie Raum an Bedeutung. DAIDALUS sucht bei den Berechnungen immer nach dem nächstgelegenen Flugmanöver, dass zur geringsten Zeitdauer in einem *Separation Volume* führt. Beachtet wird dabei lediglich der Zeitraum der Lookahead Time. Wenn die Lookahead Time entsprechend kurz gewählt wird, sollte das Ownship für längere Zeit im Flugkorridor geradeaus fliegen, da innerhalb der Lookahead Time kein Unterschied zwischen dem Kurs eines Ausweichmanövers und dem Geradeausflug besteht. Abbildung 4.19 veranschaulicht diese Überlegung.

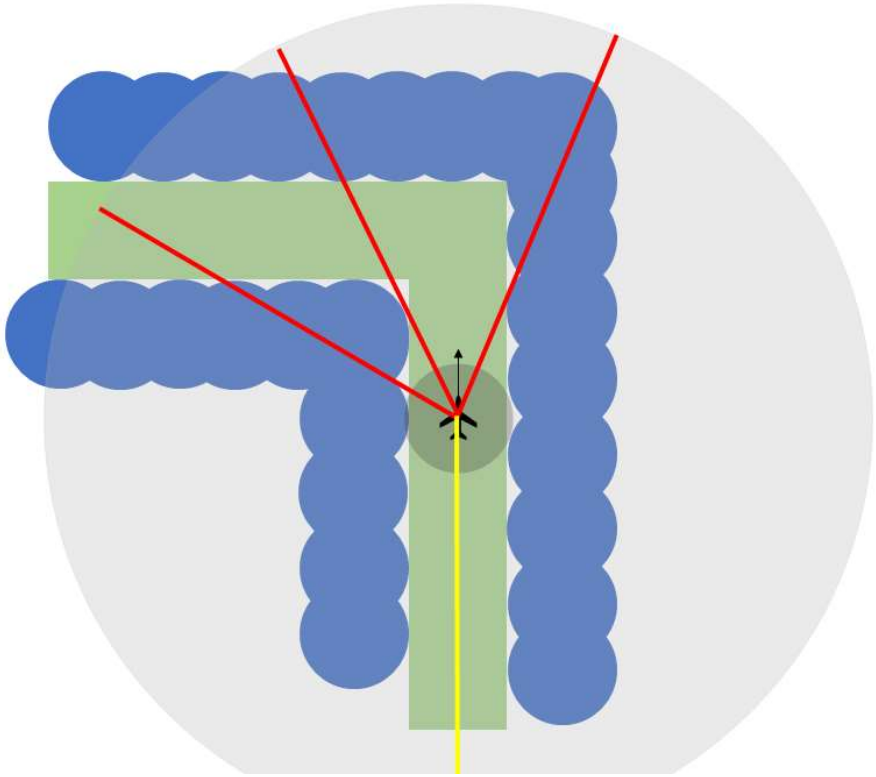


Abbildung 4.19 Einfluss der Lookahead Time

Der große hellgraue Kreis stellt den Zeithorizont dar, der mit der wie in der DO-365 (s. Tabelle 2.1) definierten Lookahead Time von $T_{DO-365} = 180\,\mathrm{s}$ durch DAIDALUS beachtet wird. Die roten Linien stellen mögliche Flugpfade dar. Innerhalb der Lookahead Time von $T_{DO-365} = 180\,\mathrm{s}$ schneidet jeder dieser möglichen Pfade mindestens ein *Separation Volume*. Die einzige Möglichkeit, innerhalb der Lookahead Time keines der Volumen zu verletzen

ist eine $180\,°$-Wende, um dem Flugkorridor in die entgegengesetzte Richtung zu folgen. Dieser Flugpfad ist in der Abbildung durch die gelbe Linie dargestellt.
Wird jedoch eine Lookahead Time von $T_{kurz} = 20\,\mathrm{s}$ (dargestellt durch den kleineren, dunkelgrauen Kreis) gewählt, so verletzen auch die zuvor kritischen Flugpfade (rote Linien) keines der *Separation Volumes*. Da somit auch der aktuelle Geradeausflug keine Warnung auslöst, wird der aktuelle Kurs beibehalten. Der Flugkorridor hat wie oben berechnet eine Breite von etwa $B = 560\,\mathrm{m}$. Das Ownship fliegt mit einer Fluggeschwindigkeit von $v_A = 25\,\mathrm{m/s}$ und damit innerhalb der Lookahead Time:

$$d = v_A \cdot T = 25\,\mathrm{m/s} \cdot 20\,\mathrm{s} = 500\,\mathrm{m}$$

Damit kann sich das Ownship innerhalb des Flugkorridors aufhalten, dort jedoch nicht wenden, weil ein Wendemanöver mit dem minimalen Wenderadius von $r_{w,min} = 136,6\,\mathrm{m}$ unmittelbar in einer Warnung resultieren würde. Stattdessen wird der aktuelle Kurs beibehalten, bis der Zeithorizont der Lookahead Time auf eines der statischen Hindernisse vor dem Ownship stößt. An dieser Stelle ist die dem aktuellen Kurs nächstgelegene Flugbahn ohne Verletzung eines *Separation Volumes* durch eine Linkskurve um das Eck zu erreichen.
Nachfolgende Daten wurden mit einer Lookahead Time von $T_{kurz} = 20\,\mathrm{s}$ gewonnen. Nachfolgende Abbildung 4.20 zeigt den gesamten Flug mit einer Flugdauer von $t_{Flugdauer} = 2:30\,\mathrm{h}$.

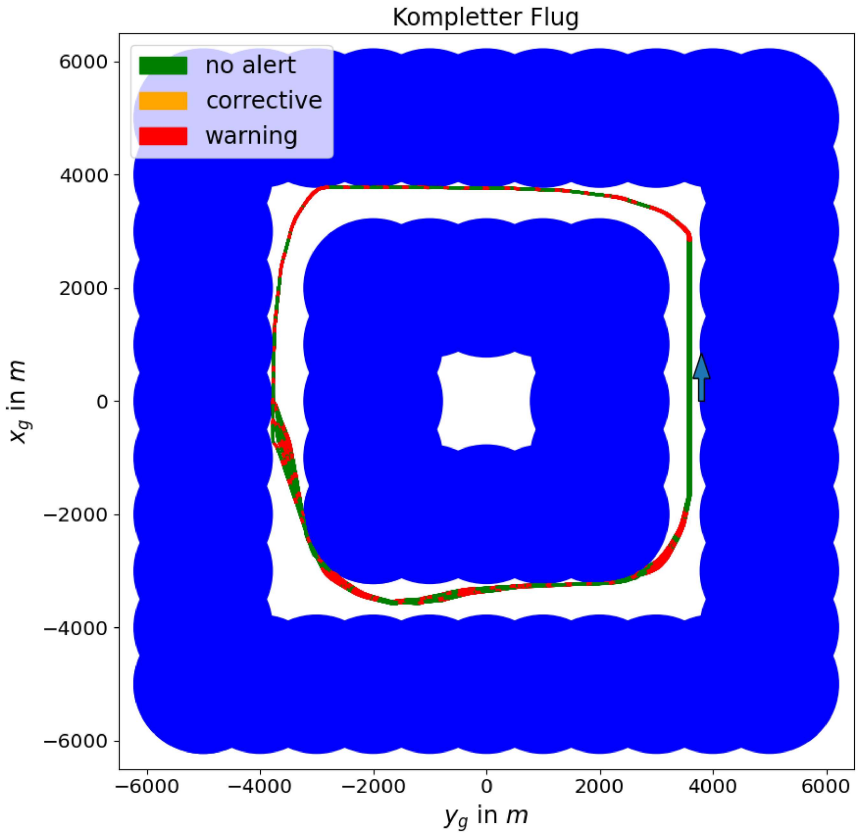


Abbildung 4.20 Gesamter Flug Szenario Flugkorridor mit verkürzter Lookahead Time

Abgesehen von der geänderten Lookahead Time wurde nichts gegenüber dem oben beschriebenen Szenario verändert. Das Ownship startet den Flug bei ($x_g = 0\,\mathrm{m}; y_g = 3500\,\mathrm{m}$) mit einem Kurs von $\chi = 360\,^\circ$ und einer Fluggeschwindigkeit von $v_A = 25\,\mathrm{m/s}$. In der Abbildung ist zu sehen, dass das Ownship mehrere Runden innerhalb des erstellten Flugkorridors fliegt. Dabei wird zu keinem Zeitpunkt eines der *Separation Volumes* verletzt. Das Ergebnis bestätigt die oben vorgestellte Vermutung.

4.2. Bewegte Hindernisse

Im Anwendungsfall des ALAADy-Demonstrators kommt es nicht nur zu Störungen durch statische Hindernisse. Auch bewegte Hindernisse - etwa (motorlose) Segelflugzeuge oder Rettungshubschrauber - können in einer Höhe bis $120\,\mathrm{m\,AGL}$ den vor dem Flug geplanten Flugpfad des ALAADy-Demonstrators kreuzen. Entsprechend wird in diesem Unterkapitel überprüft, inwiefern DAIDALUS genutzt werden kann, um bewegten Hindernissen auszuweichen. Dazu wird in Abschnitt 4.2.1 zunächst betrachtet, wie zuverlässig DAIDALUS einem frontal entgegenkommenden Luftfahrzeug ausweicht. In einem dieser Arbeit vorangegangenen Praktikum beim DLR in Braunschweig waren Pascal Guillouet bei der Entwicklung eines DAA Algorithmus nach einem geometrischen Ansatz ein Szenario aufgefallen, in dem es zu Problemen kam. Dieses Szenario wird in Abschnitt 4.2.2 betrachtet. Es wird ein von oben auf das Ownship absinkendes Luftfahrzeug betrachtet. Um das Unterkapitel der bewegten Hindernisse abzuschließen, wird in Abschnitt 4.2.3 ein bewegtes Hindernis aus unterschiedlichen Richtungen und mit unterschiedlichen Geschwindigkeiten betrachtet.

4.2.1. Frontal entgegenkommendes Luftfahrzeug

In diesem Abschnitt wird ein fremdes Luftfahrzeug betrachtet, dass sich dem Ownship frontal nähert. Dargestellt ist dieses Szenario schematisch in Abbildung 4.21.

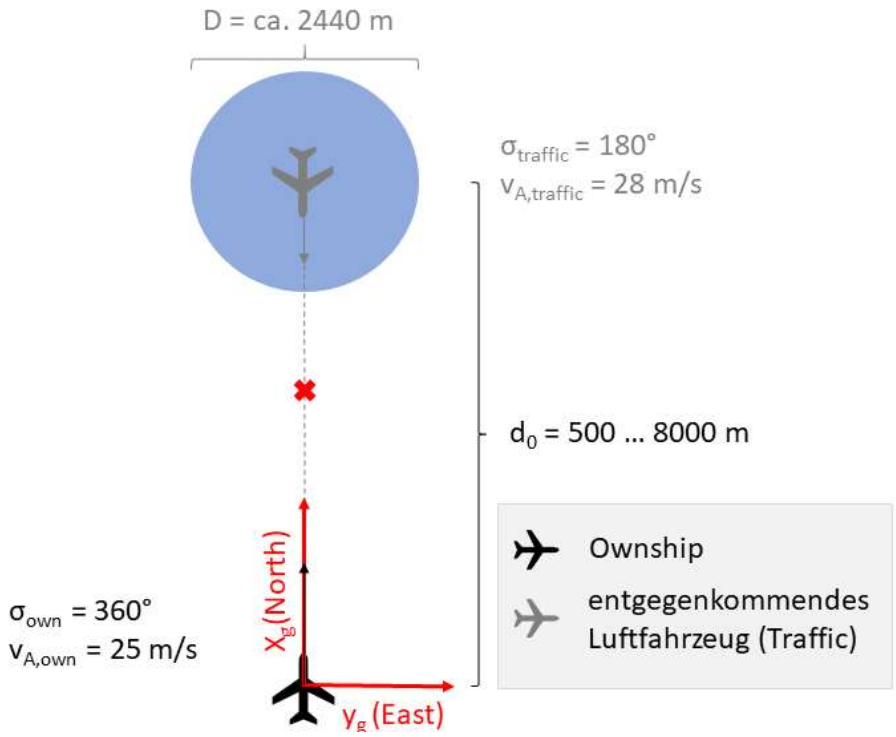


Abbildung 4.21 Schematische Darstellung Szenario frontal entgegenkommendes Luftfahrzeug

In diesem Szenario startet das Ownship im Ursprung des Koordinatensystems mit einem Kurs von $\chi_{own} = 360\,^\circ$ und einer konstanten Geschwindigkeit von $v_{A,own} = 25\,\mathrm{m/s}$. Das entgegenkommende Luftfahrzeug startet in einer Entfernung von $d_0 = 500\,\mathrm{m}$ bis $d_0 = 8000\,\mathrm{m}$ zum Ownship ($x_{g,traffic} = 500...8000\,\mathrm{m}$). Die Entfernung wird pro Berechnung um $500\,\mathrm{m}$ erhöht. Das entgegenkommende Luftfahrzeug fliegt mit einer Geschwindigkeit von $v_{A,traffic} = 28\,\mathrm{m/s}$[4] und einem Kurs von $\chi_{traffic} = 180\,^\circ$. Das entgegenkommende Luftfahrzeug hält Geschwindigkeit und Kurs konstant. Wird vom Ownship kein Ausweichmanöver geflogen, so kollidieren beide Luftfahrzeuge - in Abbildung 4.21 durch ein rotes Kreuz angedeutet.
Im Folgenden werden die von DAIDALUS in diesem Szenario empfohlenen Ausweichmanöver betrachtet. Abbildung 4.22 zeigt alle Flugpfade übereinandergelegt. Der blaue Pfeil stellt die Ausgangsflugrichtung dar.

[4] ca. 100km/h

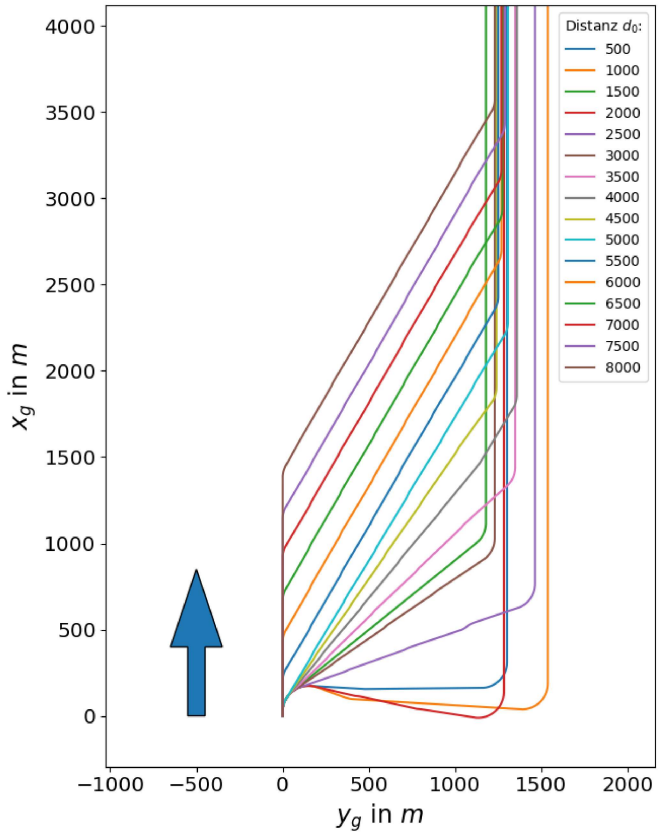
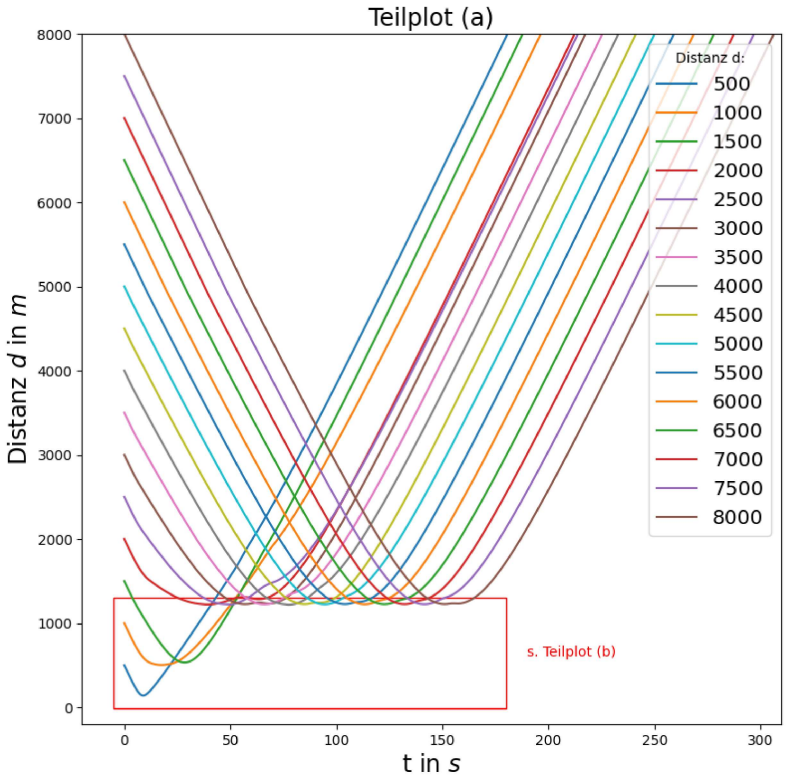


Abbildung 4.22 Flugpfade Szenario frontal entgegenkommendes Luftfahrzeug

Es ist zu erkennen, dass in allen berechneten Flügen ein Ausweichmanöver nach rechts geflogen wird. Auffällig ist, dass das Ausweichmanöver mit zunehmender Anfangsdistanz d_0 später eingeleitet wird. Dass zu Beginn eine größere Distanz zwischen beiden Luftfahrzeugen liegt und ein Ausweichmanöver damit nicht unverzüglich zu Beginn der Berechnung nötig ist, erklärt dieses Verhalten. Außerdem zu beachten sind die Flüge mit einer Anfangsdistanz von $d_{500} = 500\,\mathrm{m}$ bzw. $d_{1000} = 1000\,\mathrm{m}$. Für d_{500} ist das Auftreten einer NMAC unumgänglich, da der Radius des *NMAC Volumes* wie in Tabelle 2.1 aufgeführt auf $DTHR_{NMAC} = 500\,\mathrm{m}$ festgelegt ist. Da sich beide Luftfahrzeuge frontal entgegenkommen, wird das *NMAC Volume* damit bereits bei $t = 0,1\,\mathrm{s}$ verletzt. Ähnliches gilt für d_{1000}. Hier startet die Berechnung bereits innerhalb des *Preventive/Corrective/Recovery Volumes*. Dennoch berechnet DAIDALUS für beide Fälle ein Ausweichmanöver nach rechts, um die Zeitdauer, die das Ownship in den jeweiligen Volumen verbringt, zu minimieren. Bei einer genaueren Betrachtung ist zu erkennen, dass die übrigen Flüge (Anfangsdistanz d_0 zwischen $1500\,\mathrm{m}$ und $8000\,\mathrm{m}$) zwischen $y_g = 1180\,\mathrm{m}$ und $y_g = 1465\,\mathrm{m}$ auf den ursprünglichen Kurs $\chi = 360\,^\circ$ zurückschwenken. Dabei ist kein direkter Zusammenhang zwischen der Anfangsdistanz d_0 und der y_g-Koordinate zum Ende des Fluges gegeben. Die y_g-Koordinate zum Ende des Fluges hängt von vielen Faktoren ab. Beispielsweise von dem Zeitpunkt, an dem nach dem Ausweichmanöver keine Warnung mehr von DAIDALUS ausgegeben wird. Oder vom Aktualisierungszeitpunkt der DAIDALUS Berechnung sowie der Hysterese des DAIDALUS Algorithmus. Eine eindeutige Erklärung der unterschiedlichen y_g-Koordinaten am Ende der Flüge ist daher nicht möglich.

Entscheidend für die Bewertung eines DAA Algorithmus ist die geringste Distanz, die zu den Hindernissen eingehalten wird. Abbildung 4.23 zeigt in Teilplot (a) die Distanz der einzelnen Flüge, geplotet über die Zeit. In Teilplot (b) ist eine Vergrößerung des in Teilplot (a) markierten Bereichs.



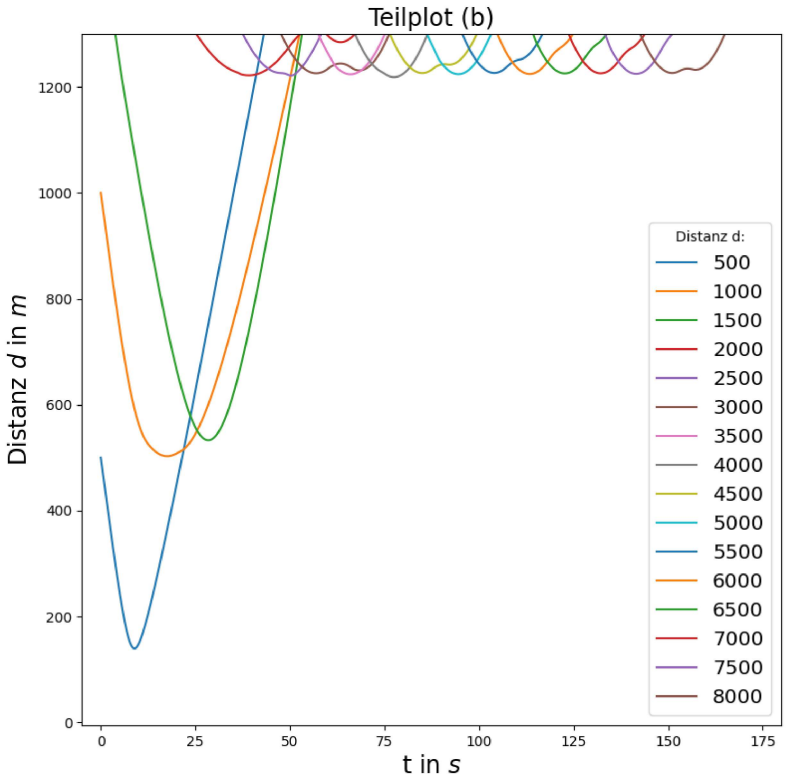


Abbildung 4.23 Distanz Szenario frontal entgegenkommendes Luftfahrzeug

In Teilplot (a) ist zu erkennen, dass die Distanz zu Beginn des Fluges für große Distanzen nahezu linear abnimmt. Nachdem das Ownship das Ausweichmanöver eingeleitet hat, flacht die Kurve bis zu ihrem Minimum ab und steigt anschließend wieder. Der Bereich der Minima ist in Teilplot (b) vergrößert dargestellt. Bei Betrachtung von Teilplot (b) fällt auf, dass bei dem Flug mit einer Anfangsdistanz von $d_0 = 500\,\mathrm{m}$ eine *NMAC* auftritt. Die geringsten Distanzen liegt dabei bei $d_{500,min} = 139,7\,\mathrm{m}$. Bei einer Anfangsdistanz von $d_0 \geq 1000\,\mathrm{m}$ wird eine *NMAC* erfolgreich verhindert. Bei einer Anfangsdistanz von $d_0 \geq 2000\,\mathrm{m}$ wird auch eine Verletzung des Corrective Volumes des fremden Luftfahrzeugs verhindert. Die geringste Distanz liegt für Flüge mit einer Anfangsdistanz von $d_0 \geq 2000\,\mathrm{m}$ bei $d_{2500,min} = 1221,1\,\mathrm{m}$.

4.2.2. Von oben absinkendes Luftfahrzeug

Pascal Guillouet erarbeitete in seinem dieser Arbeit vorangegangenen Praktikum ein Problem des von ihm herausgearbeiteten DAA Algorithmus. Bei einem von oben auf das Ownship absinkenden Luftfahrzeug kommt es in Guillouets Algorithmus zu einer *NMAC*. Begründet wird dieses Ergebnis durch zwei existierende Lösungen in seinem Algorithmus, wovon zum Teil die ungünstigere ausgewählt wird. Abbildung 4.24 zeigt eine Skizze dieses Szenarios.

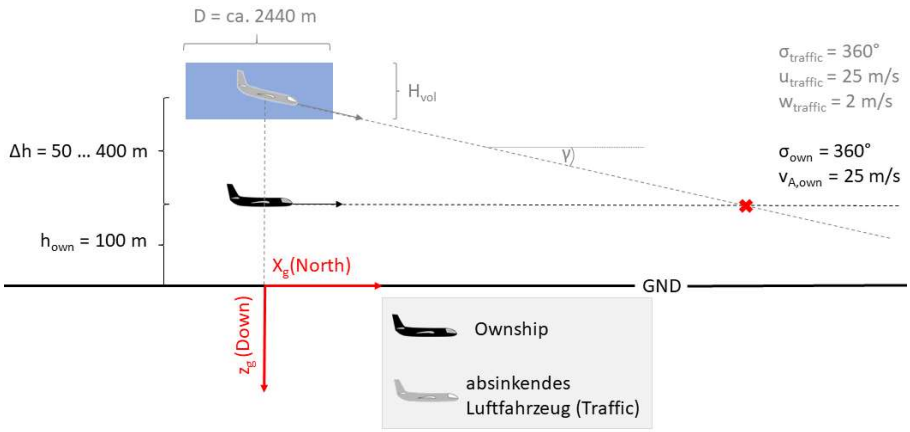


Abbildung 4.24 Schematische Darstellung Szenario von oben auf Ownship absinkendes Luftfahrzeug

In diesem Szenario startet das Ownship mit einer Höhe von $h_{own} = 100\,\mathrm{m}$ über dem Ursprung des Koordinatensystems (d. h. $z_{g,own} = -100\,\mathrm{m}$). Das Ownship befindet sich in einem horizontalen Flug mit einem Kurs von $\chi_{own} = 360\,^\circ$. Die Fluggeschwindigkeit beträgt $v_{A,own} = 25\,\mathrm{m/s}$ und bleibt über den gesamten Flug konstant. Das von oben auf das Ownship absinkende Luftfahrzeug startet mit einer Höhendifferenz von $\Delta\,h_1 = 50\,\mathrm{m}$ bis $\Delta\,h_8 = 400\,\mathrm{m}$ über dem Ownship. Die z_g-Koordinate des absinkenden Ownships beträgt damit zu Beginn des Fluges $z_{g,1} = -150\,\mathrm{m}$ bis $z_{g,8} = -500\,\mathrm{m}$. Das absinkende Luftfahrzeug fliegt mit einer Geschwindigkeit von $u_{traffic} = 25\,\mathrm{m/s}$ mit Kurs $\chi_{traffic} = 360\,^\circ$. Die Vertikalgeschwindigkeit des absinkenden Luftfahrzeugs beträgt $w_{traffic} = 2\,\mathrm{m/s}$. Dadurch ergibt sich ein Flugbahnneigungswinkel von

$$\gamma = \arctan\left(\frac{w_{traffic}}{u_{traffic}}\right) = \arctan\left(\frac{2\,\mathrm{m/s}}{25\,\mathrm{m/s}}\right) = -4,6\,^\circ$$

Die Höhe des *Preventive Volumes* (vom Mittelpunkt des Luftfahrzeugs gemessen) beträgt gemäß Tabelle 2.1 ca. $ZTHR_{prev} = 215\,\mathrm{m}$, beziehungsweise für das *Corrective Volume* und das *Recovery Volume* $ZTHR_{cor} = ZTHR_{rec} = 140\,\mathrm{m}$. Das *NMAC Volume* ist mit $ZTHR_{NMAC} = 20\,\mathrm{m}$ definiert. Abbildung 4.25 zeigt die in diesem Szenario geflogenen Flugpfade des Ownships. Dabei ist der Flug mit einer Höhendifferenz von $\Delta\,h_{50} = 50\,\mathrm{m}$ in den x_g- und y_g-Koordinaten deckungsgleich mit den Flügen einer Höhendifferenz von $\Delta\,h_{100} = 100\,\mathrm{m}$, $\Delta\,h_{150} = 150\,\mathrm{m}$ und $\Delta\,h_{200} = 200\,\mathrm{m}$.

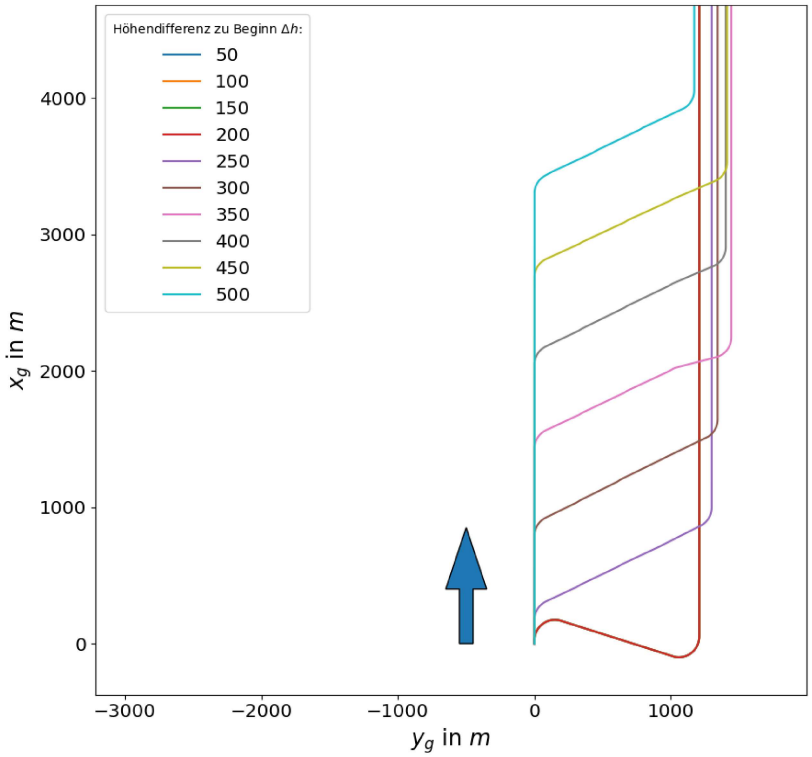


Abbildung 4.25 Flugpfade Szenario von oben auf Ownship absinkendes Luftfahrzeug

Es ist zu sehen, dass jeder der simulierten Flüge ein Ausweichmanöver nach rechts vornimmt. Je größer die Höhendifferenz zu Beginn ist, desto später wird dieses Ausweichmanöver eingeleitet. Die oben beschriebenen deckungsgleichen Flüge starten alle mit einer Höhendifferenz $\Delta h < 215\,\mathrm{m}$ und damit mindestens innerhalb des *Preventive Volumes*. Die vertikale Zeitkonstante $TCOA$ ist für alle Volumen auf $TCOA = 0\,\mathrm{s}$ festgelegt. Die Zeitkonstante wird erst mit Erreichen derselben Höhe verletzt und muss nicht weiter beachtet werden, da $ZTHR$ bereits vor $TCOA$ verletzt wird. Obwohl das *Recovery Volume* erst ab einer Höhendifferenz $\Delta h < 120\,\mathrm{m}$ verletzt wird, wird bei den Flügen bis zu einer Höhendifferenz zu Beginn des Fluges von $\Delta h_0 \leq 150\,\mathrm{m}$ bereits zu Beginn des Fluges eine *Warning* vorzuweisen. Der Flug mit einer Höhendifferenz zu Beginn des Fluges von $\Delta h_0 = 200\,\mathrm{m}$ weist zu Beginn einen *Corrective Alert* auf. Alle anderen berechneten Flüge starten ohne eine Warnung von DAIDALUS.
Abbildung 4.26 zeigt zwei Teilplots. Teilplot(a) zeigt den Betrag der horizontalen Distanz zwischen dem Ownship und dem absinkenden Luftfahrzeug über die Zeit t. Teilplot (b) zeigt den Betrag der vertikalen Distanz zwischen beiden Luftfahrzeugen über die Zeit t.

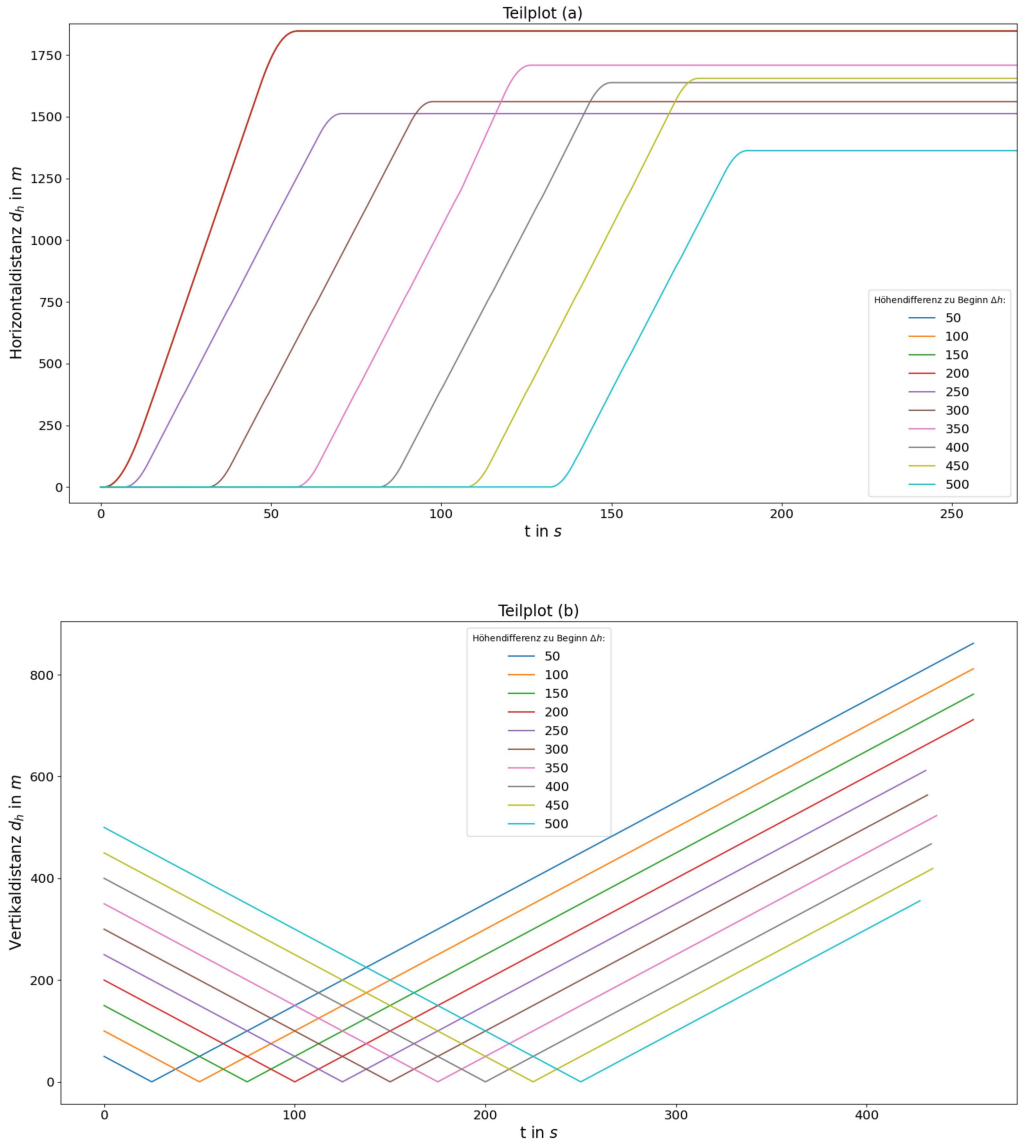


Abbildung 4.26 Distanzen Szenario von oben auf Ownship absinkendes Luftfahrzeug

Teilplot (a) zeigt eine über die Zeit stetig wachsende horizontale Distanz, bis eine Distanz zwischen $d_h = 1360\,\mathrm{m}$ und $d_h = 1850\,\mathrm{m}$ erreicht ist. Das Verhalten ist nach Betrachtung der Ausweichmanöver in Abbildung 4.25 zu erwarten. Teilplot (b) in Abbildung 4.26 zeigt eine linear sinkende vertikale Distanz d_v, bevor sie - nachdem beide Luftfahrzeuge auf derselben Höhe sind - wieder ansteigt. Durch die konstante Vertikalgeschwindigkeit $w_{traffic}$ des absinkenden Luftfahrzeugs und die konstant gehaltene Höhe des Ownships ist das lineare Verhalten zu erklären. Abbildung 4.27 zeigt die Kombination beider Distanzplots. Dabei bietet Teilplot (a) in Abbildung 4.27 eine Übersicht und Teilplot (b) einen Ausschnitt von Teilplot (a).

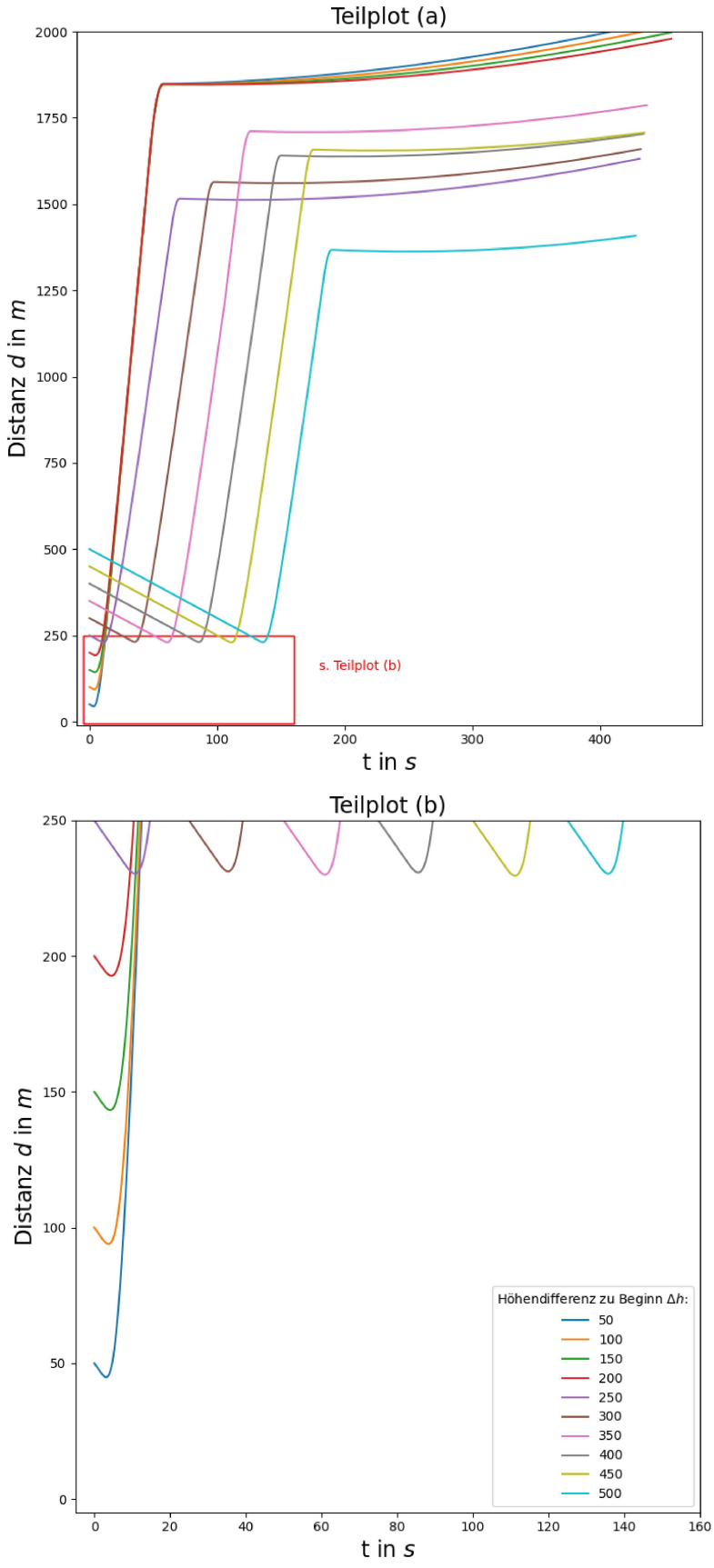


Abbildung 4.27 Gesamtdistanz Szenario von oben auf Ownship absinkendes Luftfahrzeug

In Teilplot (a) ist zu erkennen, dass die direkte Distanz d zwischen beiden Luftfahrzeugen ebenfalls linear abnimmt, bevor die Abnahme nichtlinear gestoppt wird und die direkte Distanz nachfolgend wieder deutlich steigt. In Teilplot (b) ist zu erkennen, dass die geringste Distanz beim Flug mit der Höhendifferenz Δh_{50} bei $d_{min,50} = 44,8\,\mathrm{m}$ liegt. Die minimale Distanz ist bei $t = 3,1\,\mathrm{s}$ erreicht. Bei $t = 3,1\,\mathrm{s}$ liegt die vertikale Distanz in diesem Flug bei $d_v = 43,8\,\mathrm{m}$, die horizontale Distanz beträgt $d_h = 9,5\,\mathrm{m}$. Durch die vertikale Distanz ist sichergestellt, dass eine *NMAC* verhindert wird. Eine *NMAC* liegt vor, wenn die vertikale Distanz $\Delta h < 20\,\mathrm{m}$ beträgt. Durch dieselbe Analyse zeigt sich, dass auch in allen anderen simulierten Flügen dieses Szenarios eine *NMAC* erfolgreich verhindert wird. Ab einer Höhendifferenz zu Beginn des Fluges von $\Delta h \geq 200\,\mathrm{m}$ wird das *Preventive Volume* nicht mehr verletzt.
In weitergehenden Versuchen wird die Höhendifferenz zu Beginn des Fluges in $2\,\mathrm{m}$-Schritten von $\Delta h = 50\,\mathrm{m}$ bis auf $\Delta h = 20\,\mathrm{m}$ reduziert. Eine *NMAC* tritt auf, sofern die Höhendifferenz zu Beginn des Fluges weniger als $\Delta h = 26\,\mathrm{m}$ beträgt. Bei einer Höhendifferenz zu Beginn des Fluges von $\Delta h \geq 50\,\mathrm{m}$ wird erfolgreich die Verletzung des *Corrective Volumes* und des *Recovery Volumes* verhindert.

4.2.3. Luftfahrzeug aus verschiedenen Richtungen

Dieser Abschnitt schließt das Unterkapitel der bewegten Hindernisse. Pro simuliertem Flug wird ein Luftfahrzeug auf Kollisionskurs mit dem Ownship gebracht. Abbildung 4.28 zeigt eine schematische Darstellung dieser Versuchsreihe.

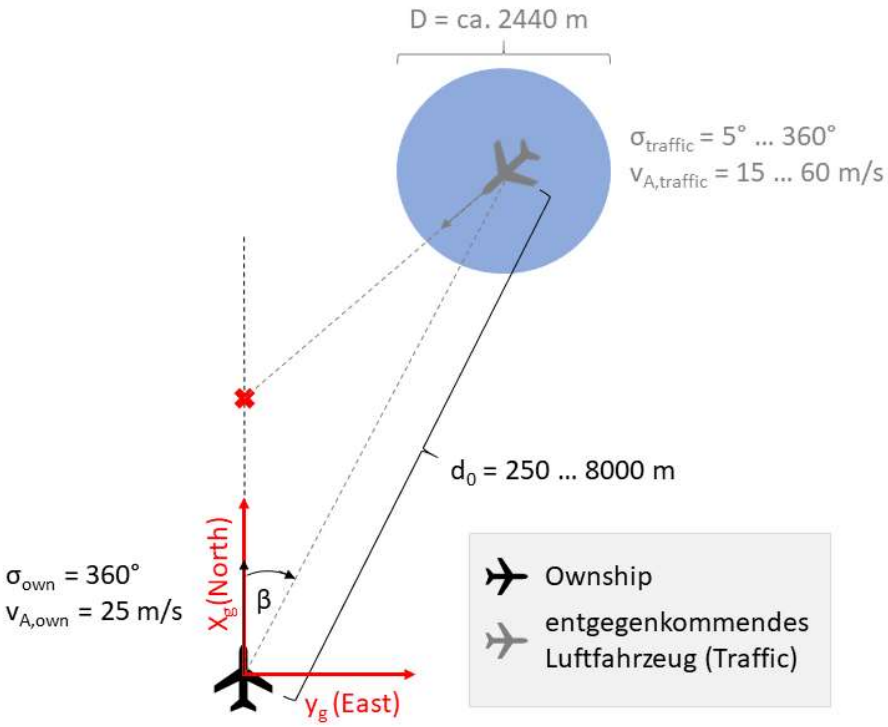


Abbildung 4.28 Schematische Darstellung Szenario Luftfahrzeug aus verschiedenen Richtungen

Das Ownship beginnt jeden Flug im Ursprung des Koordinatensystems mit einer Fluggeschwindigkeit von $v_{A,own} = 25\,\mathrm{m/s}$ und einem Kurs von $\chi = 360\,°$. Das fremde Luftfahrzeug wird in einer Entfernung von $d_0 = 250...8000\,\mathrm{m}$ so zum Ownship orientiert, dass es zu einer

Kollision auf der x_g-Achse kommt, sollte vom Ownship kein Ausweichmanöver eingeleitet werden. Der Kollisionspunkt ist in Abbildung 4.28 als rotes Kreuz gekennzeichnet. Der Kurs des fremden Luftfahrzeugs wird dabei von $\chi_{traffic} = 5\,^\circ$ in $5\,^\circ$-Schritten bis auf $\chi_{traffic} = 360\,^\circ$ erhöht. Zudem wird die Fluggeschwindigkeit des fremden Luftfahrzeugs variiert. Hier werden die Werte $v_{A,traffic} = [15^5; 25^6; 30^7; 40^8; 50^9; 60^{10}\,]/\mathrm{m}s$ verwendet. Dadurch ergibt sich eine Anzahl von 72 Flügen pro Anfangsdistanz, welche jeweils mit 6 verschiedenen Fluggeschwindigkeiten des fremden Luftfahrzeugs simuliert werden. Mit den 32 unterschiedlichen Anfangsdistanzen ergibt sich eine Gesamtanzahl von $72 * 6 * 32 = 13.824$ simulierten Flügen für dieses Szenario.
Die Startposition des fremden Luftfahrzeugs ist definiert durch die Anfangsdistanz d_0 sowie den in Abbildung 4.28 eingezeichneten Winkel β:

$$\begin{pmatrix} x_{g_{0,traffic}} \\ \\ y_{g_{0,traffic}} \end{pmatrix} = \begin{pmatrix} \cos(\beta) * d \\ \\ \sin(\beta) * d \end{pmatrix} \tag{4.1}$$

Der Winkel β ist definiert durch:

$$\beta = \arctan\left(\frac{\sin(\chi_{traffic})}{\cos(\chi_{traffic}) - \frac{v_{A,own}}{v_{A,traffic}}}\right) \tag{4.2}$$

Für einen Kurs des fremden Luftfahrzeugs von $\chi_{traffic} = 0\,^\circ = 360\,^\circ$ kommt es durch den $\cos(\chi_{traffic})$ im Nenner von Gleichung 4.2 zu einer Definitionslücke (Division durch 0), wenn Ownship und Traffic mit derselben Fluggeschwindigkeit ($v_A = 25\,\mathrm{m/s}$) fliegen. In diesem Fall startet das fremde Luftfahrzeug bei ($x_g = d_0; y_g = 0$).
Abbildung 4.29 zeigt Beispielhaft die ersten Sekunden des fremden Luftfahrzeugs für die Flüge mit einer Fluggeschwindigkeit von $v_{A,traffic} = 60\,\mathrm{m/s}$ und einer Anfangsdistanz von $d_0 = 5000\,\mathrm{m}$.

[5] $54\,\mathrm{km/h}$
[6] $90\,\mathrm{km/h}$
[7] $108\,\mathrm{km/h}$
[8] $144\,\mathrm{km/h}$
[9] $180\,\mathrm{km/h}$
[10] $216\,\mathrm{km/h}$

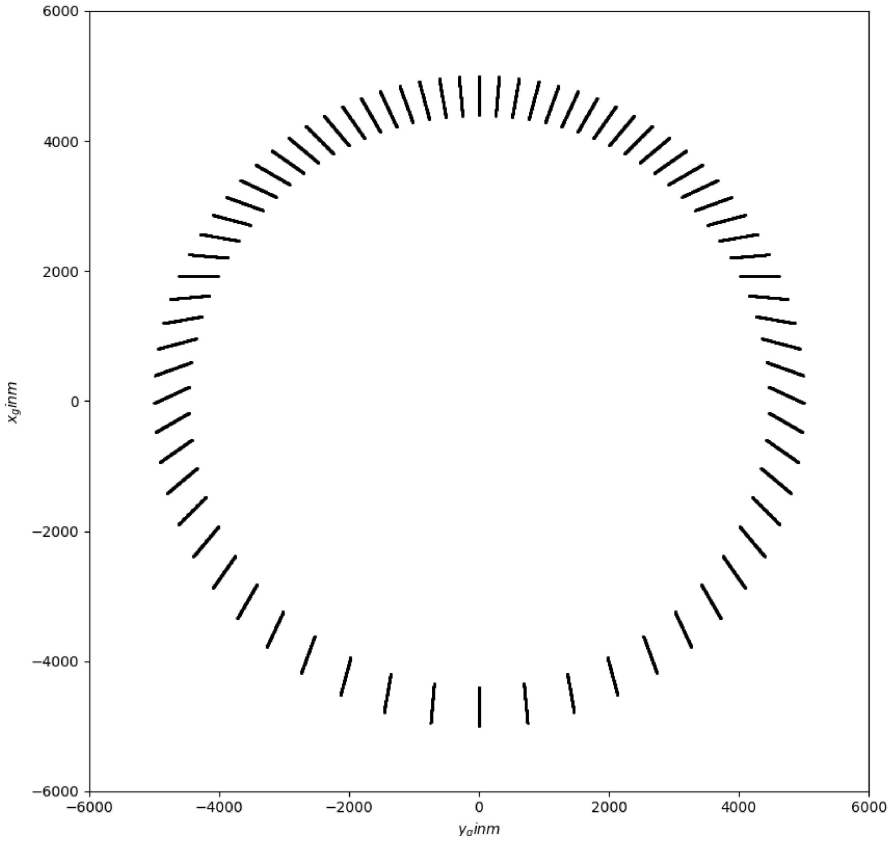


Abbildung 4.29 Startpositionen des fremden Luftfahrzeugs bei $v_{A,traffic} = 60\,\mathrm{m/s}$ und $d_0 = 5000\,\mathrm{m}$

Deutlich zu erkennen ist die kreisrunde Anordnung der Startpunkte, die sich durch die $\cos()$-$\sin()$-Beziehung der Startkoordinaten und die feset definierte Distanz zum Ownship ergibt. Ebenso ist die Orientierung der Flugrichtungen in positive x_g-Richtung zu erkennen. Der Kollisionspunkt liegt jeweils auf der x_g-Achse. Die y_g-Koordinate unterscheidet sich je nach Anfangsdistanz d_0, der Position relativ zum Ownship sowie der Relativgeschwindigkeit zum Ownship. Folgende Fälle werden betrachtet:

- Fall (a): $v_{A,own} < v_{A,traffic}$
- Fall (b): $v_{A,own} = v_{A,traffic}$
- Fall (c): $v_{A,own} > v_{A,traffic}$

Fall (a): $v_{A,own} < v_{A,traffic}$

Fall (a) bezieht sich auf die Flüge des fremden Luftfahrzeugs mit einer Geschwindigkeit von $v_{Atraffic} = [30; 40; 50; 60\,\mathrm{m/s}]$. Die Auswertung der Daten lässt Rückschlüsse darauf ziehen, bei welcher Entfernung ein fremdes Luftfahrzeug vom Ownship erkannt werden muss, um diesem noch rechtzeitig ausweichen zu können. Abbildung 4.30 zeigt eine solche Analyse für eine Fluggeschwindigkeit des fremden Luftfahrzeugs von $v_{A,traffic} = 60\,\mathrm{m/s}$. Die entsprechenden Abbildungen der übrigen Flüge dieses Szenarios sind im Anhang B zu finden. Die Winkelangabe gibt an, aus welcher Richtung ϵ das fremde Luftfahrzeug kommt. Das bedeutet $\epsilon = \chi_{traffic} - 180\,°$. Die in rot dargestellten Werte repräsentieren die Entfernung, die bei

der jeweiligen Richtung mindestens notwendig ist, um eine *NMAC* zu vermeiden. Die blauen Werte geben an, bei welcher Entfernung die Verletzung eines *Separation Volumes* verhindert wird. Bei allen Flügen mit einer größeren Anfangsdistanz als dem jeweiligen Wert wird das entsprechende Volumen nicht verletzt.

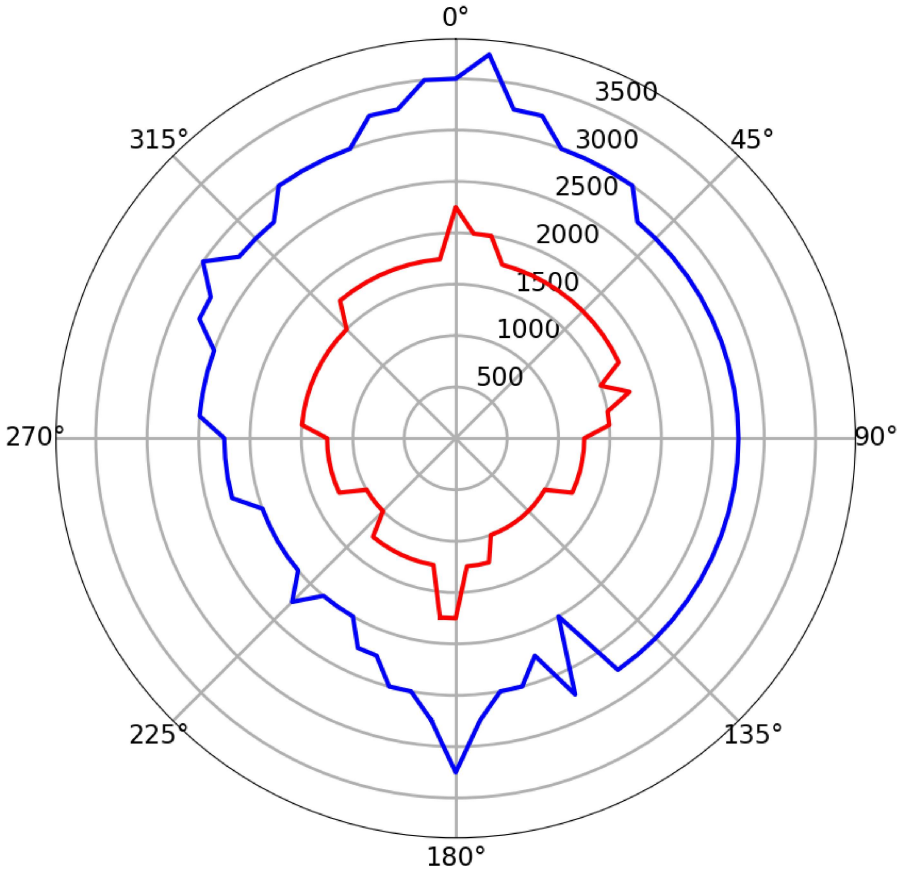


Abbildung 4.30 Mindeststartdistanzen Fall (a): $v_{A,own} < v_{A,traffic} = 60\,\mathrm{m/s}$

In obiger Abbildung ist zu erkennen, dass ein mit $v_{A,traffic} = 60\,\mathrm{m/s}$ fliegendes Luftfahrzeug etwa zwischen $2000\,\mathrm{m}$ und $3000\,\mathrm{m}$ erkannt werden muss, damit das Ownship die *Separation Volumes* nicht verletzt. Ein Eindeutiger Verlauf dieser Grenze ist bei einer Richtung von $\epsilon = [40\,°; 145\,°]$ zu erkennen. Hier muss das fremde Luftfahrzeug eindeutig bei einer Entfernung von mindestens $d_0 = 2250\,\mathrm{m}$ vom Ownship erkannt werden. Daneben sind die Ausschläge im Bereich von $\epsilon = 0\,°$ beziehungsweise $\epsilon = 180\,°$ zu erkennen. Eine Begründung hierfür ist, dass bei einer Richtung von $\epsilon = 0\,°$ die Relativgeschwindigkeit der beiden Luftfahrzeuge bei $v_{A,rel} = 85\,\mathrm{m/s}$ liegt und die verbleibende Reaktionszeit, um ein Ausweichmanöver zu fliegen, somit sehr begrenzt ist. Auf der anderen Seite nähert sich das fremde Luftfahrzeug aus $\epsilon = 180\,°$ so schnell direkt von hinten, dass das Ownship wegen der geringen Relativgeschwindigkeit durch eine Kursänderung nur eine mäßige Änderung der Distanz erreichen kann. Für ein fremdes Luftfahrzeug aus der Richtung $\epsilon = [190\,°; 340\,°]$ ist ein schwankender Verlauf der Kurve gegeben. Es ist aber eine Tendenz erkennbar, dass ein von vorn kommendes Luftfahrzeug früher erkannt werden muss (bei einer Distanz von ca. $d_0 = 3000\,\mathrm{m}$) als ein von hinten kommendes Luftfahrzeug (bei ca. $d_0 = 2000\,\mathrm{m}$). Der Unterschied ist ebenfalls

durch die bei einem von vorn kommenden Luftfahrzeug höhere Relativgeschwindigkeit zu erklären.
Um eine *NMAC* zu verhindern, muss ein fremdes Luftfahrzeug bei den gegebenen Geschwindigkeiten spätestens bei einer Distanz von $d_0 = 2250\,\mathrm{m}$ vom Ownship erfasst werden. Auch hier sind die Spitzen um $\epsilon = 0\,^\circ$ beziehungsweise $\epsilon = 180\,^\circ$ zu sehen. Zwar schwankt auch die rote Kurve in Abbildung 4.30, jedoch deutlich geringer als die blaue Kurve. Bei der Betrachtung der anderen Abbildungen von Fall (a) (s. Anhang B) fällt eine deutlich glattere rote Kurve auf. Ein von hinten kommendes Luftfahrzeug muss (mit Ausnahme der Spitze um $\epsilon \approx 180\,^\circ$) bei deiner Entfernung von $d_0 = 750\,\mathrm{m}$ bis $d_0 = 1000\,\mathrm{m}$ erkannt werden, um eine *NMAC* zu vermeiden. Bei einem von vorn kommenden fremden Luftfahrzeug wird die *NMAC* verhindert, wenn das Luftfahrzeug bei deiner Entfernung von $d_0 = 1250\,\mathrm{m}$ bis $d_0 = 1750\,\mathrm{m}$ erkannt wird. Langsamere Luftfahrzeuge müssen eher später erkannt werden, da die Relativgeschwindigkeit der beiden Luftfahrzeuge hierbei gering genug ist, um auch mit einem später eingeleiteten Ausweichmanöver ausreichend Distanz zum Intruder halten zu können.
Werden die geringsten Distanzen aller Flüge einer einzelnen Startdistanz betrachtet, fällt eine gewisse Asymmetrie der Ergebnisse auf. Abbildung 4.31 zeigt beispielhaft alle Flüge für eine Fluggeschwindigkeit des fremden Luftfahrzeugs von $v_{A,traffic} = 60\,\mathrm{m/s}$ und eine Anfangsdistanz von $d_0 = 2250\,\mathrm{m}$.

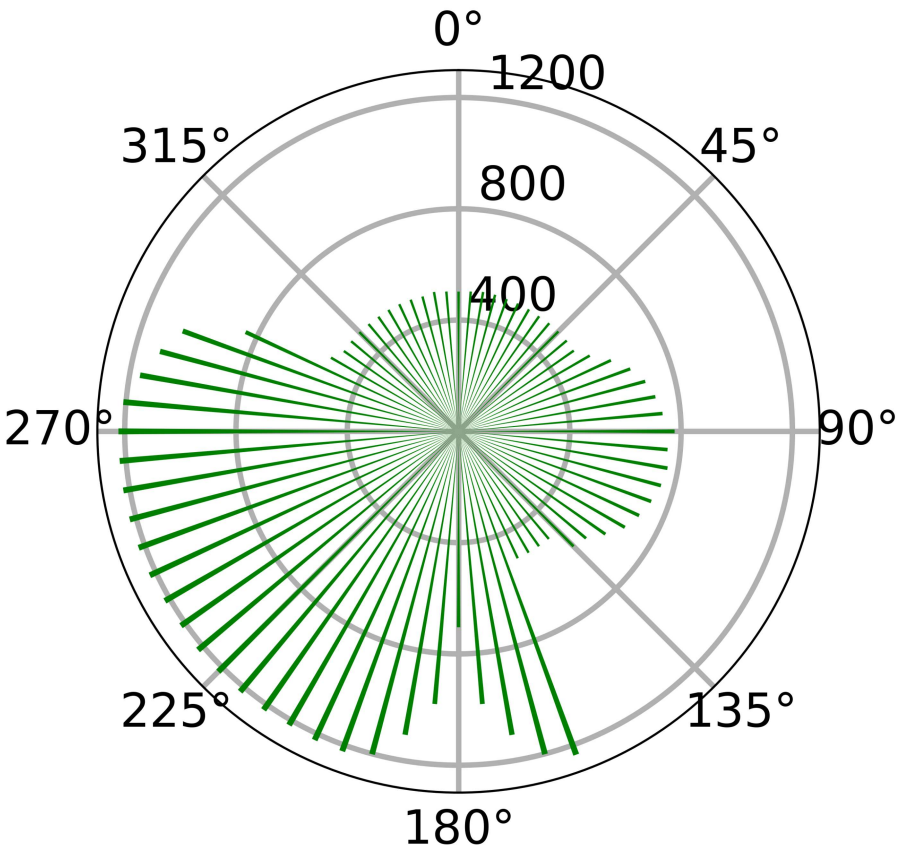


Abbildung 4.31 Minimaldistanzen für $v_{A,traffic} = 60\,\mathrm{m/s}$ und $d_0 = 2250\,\mathrm{m}$

Jeder der Balken steht für einen einzelnen Flug aus der jeweiligen Richtung. Beispielsweise wird beim Flug aus einer Richtung von $\epsilon = \chi_{traffic} - 180\,^\circ = 225\,^\circ$ zu keinem Zeitpunkt

eine Distanz von $d = 1222\,\mathrm{m}$ zum Ownship unterschritten. Die Asymmetrie der Ergebnisse lässt sich durch die Ausweichmanöver der jeweiligen Flüge erklären. Kommt das fremde Luftfahrzeug beispielsweise aus einer Richtung von $\epsilon = 270\,^\circ$ ($\chi_{traffic} = 90\,^\circ$) weicht das Ownship mit einer Rechtskurve aus und hält die Distanz zum fremden Luftfahrzeug dabei von Beginn an möglichst hoch. Einem fremden Luftfahrzeug aus $\epsilon = 90\,^\circ$ ($\chi_{traffic} = 270\,^\circ$) weicht das Ownship ebenfalls durch eine Rechtskurve aus. Während dieser Rechtskurve wird die Distanz zum fremden Luftfahrzeug entsprechend verkürzt. Die Sprünge der in der Abbildung gezeigten Minimaldistanzen sind durch geänderte Ausweichmanöver zu erklären. Beispielsweise leitet das Ownship eine Rechtskurve ein, wenn das fremde Luftfahrzeug aus einer Richtung von $\epsilon = 155\,^\circ$ ($\chi_{traffic} = 335\,^\circ$) anfliegt. Dadurch wird - ähnlich wie bei einem Anflug des fremden Luftfahrzeugs aus $\epsilon = 90\,^\circ$ ($\chi_{traffic} = 270\,^\circ$) - die Distanz bereits zu Beginn des Fluges stark verkürzt. Bei einem Anflug des fremden Luftfahrzeugs aus $\epsilon = 160\,^\circ$ ($\chi_{traffic} = 340\,^\circ$) leitet das Ownship dagegen eine Linkskurve als Ausweichmanöver ein und entfernt sich damit bestmöglich vom fremden Luftfahrzeug. Eine eindeutige Erklärung für die deutlichen Unterschiede beider Ausweichmanöver bei ähnlichen Anflugrichtungen des fremden Luftfahrzeugs kann nicht präsentiert werden. Die Asymetrie sowie die beschriebenen Sprünge der Abbildung 4.31 zeigt sich für eine Geschwindigkeit des fremden Luftfahrzeugs von $v_{A,traffic} = 60\,\mathrm{m/s}$ für alle Anfangsdistanzen $d_0 \leq 2500\,\mathrm{m}$. Für alle anderen Fluggeschwindigkeiten (auch in Fall (b) und Fall(c)) treten ähnliche Phänomene auf. Die Schwelle der Anfangsdistanz, bei der die Asymmetrie und die Sprünge nicht mehr auftreten, unterscheidet sich je nach Fluggeschwindigkeit des fremden Luftfahrzeugs. Mit abnehmender Fluggeschwindigkeit $v_{A,traffic}$ wird eine Symmetrie bei geringeren Anfangsdistanzen d_0 erreicht.

Fall (b): $v_{A,own} = v_{A,traffic}$

Abbildung 4.32 zeigt die entsprechende Analyse der Daten für ein fremdes Luftfahrzeug mit einer Fluggeschwindigkeit von $v_{A,traffic} = 25\,\mathrm{m/s}$.

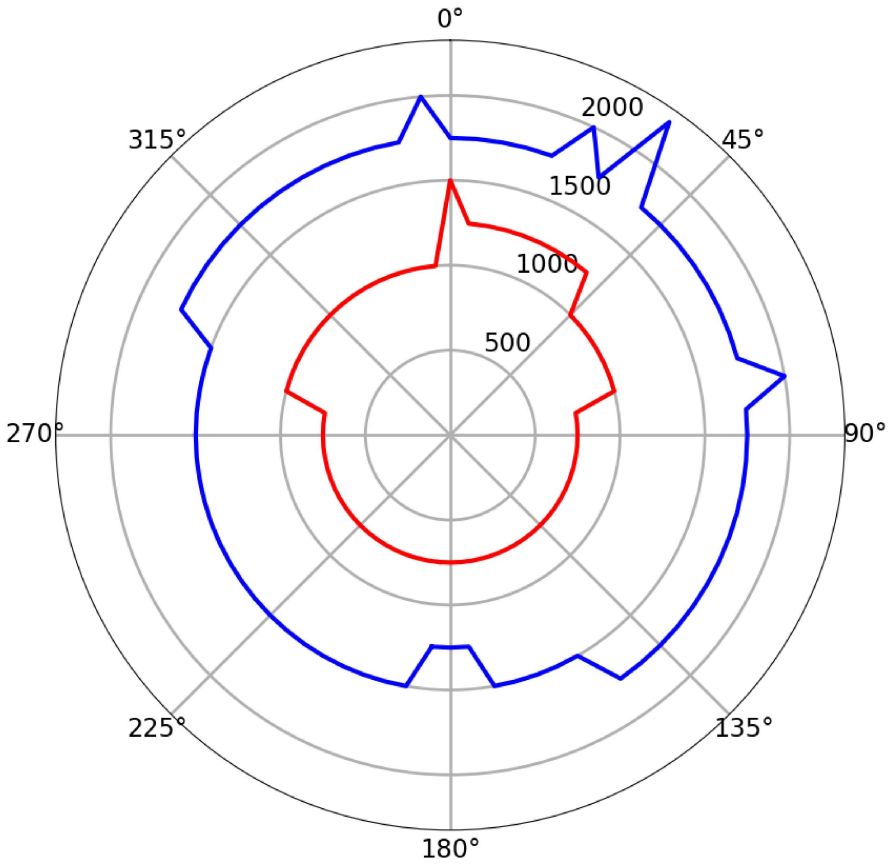


Abbildung 4.32 Mindeststartdistanzen Fall (b): $v_{A,own} = v_{A,traffic} = 25\,\mathrm{m/s}$

Im abgebildeten Plot zeigt sich ein deutlich glatteres Ergebnis als für die Flüge des fremden Luftfahrzeugs mit einer Geschwindigkeit $v_{A,traffic} > v_{A,own}$. Bis auf einige Ausnahmen bewegt sich die nötige Entfernung zum Erkennen des fremden Luftfahrzeugs für die Vermeidung einer Verletzung der *Separation Volumes* zwischen $d_0 = 1500\,\mathrm{m}$ und $d_0 = 1750\,\mathrm{m}$. Die Spitze um $\epsilon = 0\,^\circ$ fällt in diesem Fall deutlich geringer aus. Aus der anderen Richtung - $\epsilon = 180\,^\circ$ - hat sich das Verhalten sogar geändert und im Vergleich zu den umliegenden Richtungen ist eine geringere Entfernung zur Vermeidung der *Separation Volumes* nötig. Bei einem Intruder mit derselben Fluggeschwindigkeit $v_{A,traffic} = v_{A,own} = 25\,\mathrm{m/s}$ liegt die nötige Entfernung zur Vermeidung einer *NMAC* von hinten konstant bei $d_0 = 750\,\mathrm{m}$. Kommt das fremde Luftfahrzeug von vorn, so muss es spätestens ab einer Entfernung von $d_0 = 1000\,\mathrm{m}$ vom Ownship erfasst werden. Jedoch muss im Bereich von $\epsilon = 0\,^\circ$ bis $\epsilon = 40\,^\circ$ eine Entfernung von $d_0 = 1250\,\mathrm{m}$ beziehungsweise $d_0 = 1500\,\mathrm{m}$ gegeben sein.

Fall (c): $v_{A,own} > v_{A,traffic}$

Die zu einer Fluggeschwindigkeit des fremden Luftfahrzeugs von $v_{A,traffic} = 15\,\mathrm{m/s}$ gehörenden Daten sind in Abbildung 4.33 visualisiert.

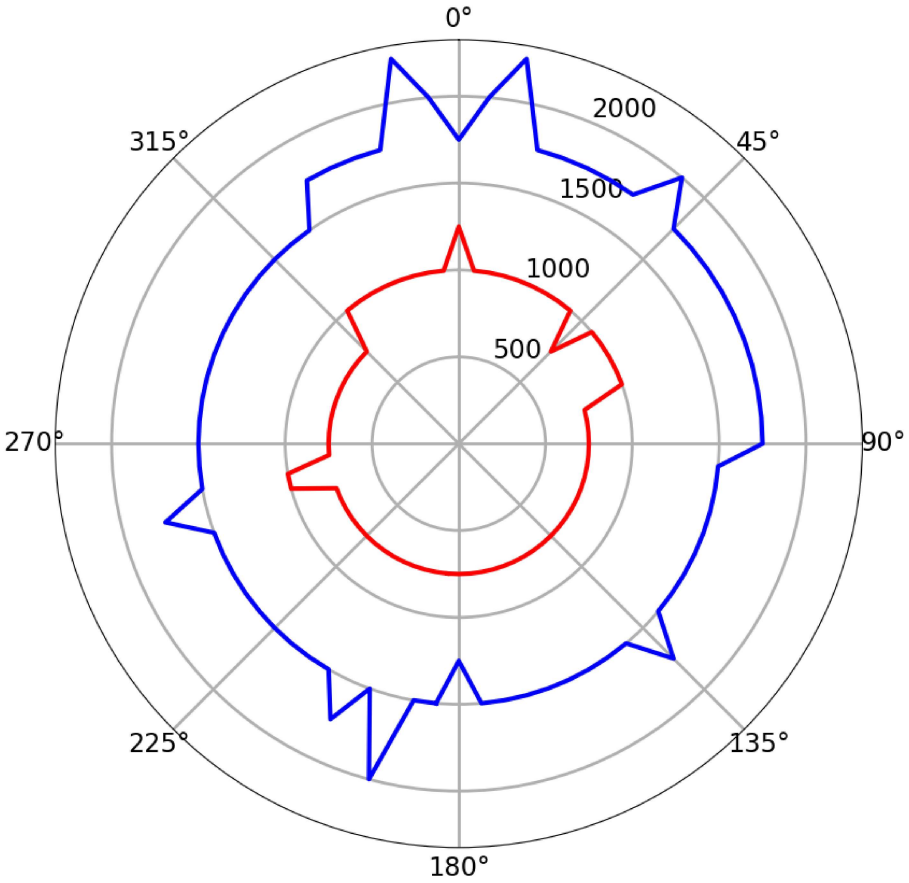


Abbildung 4.33 Mindeststartdistanzen Fall (c): $v_{A,own} > v_{A,traffic} = 15\,\mathrm{m/s}$

Bei der Betrachtung des obigen Plots fallen vor allem zwei Bereiche in den jeweiligen Kurven auf. Um einer Verletzung der *Separation Volumes* vorzubeugen, muss das fremde Luftfahrzeug aus einer Richtung von $\epsilon = [95\,^\circ; 325\,^\circ]$ in einer Entfernung von $d_0 = 1500\,\mathrm{m}$ erkannt werden. Für die Richtungen $\epsilon = [330\,^\circ; 90\,^\circ]$ muss das fremde Luftfahrzeug in der Regel bei einer Entfernung von $d_0 = 1750\,\mathrm{m}$ erkannt werden, um eine Verletzung der *Separation Volumes* zu verhindern. Eine Ausnahme stellt wiederum der Bereich um $\epsilon = 0\,^\circ$ dar. Zwar ist bei $\epsilon = 0\,^\circ$ eine Entfernung von $d_0 = 1750\,\mathrm{m}$ zur Verhinderung der Verletzung der *Separation Volumes* ausreichend, für die angrenzenden Richtungen muss das fremde Luftfahrzeug allerdings in einer Entfernung von $d_0 = 2250\,\mathrm{m}$ erkannt werden. Die rote Kurve bietet ein glatteres Verhalten. Um eine NMAC zu vermeiden, muss ein fremdes Luftfahrzeug aus einer Richtung von $\epsilon = [80\,^\circ; 315\,^\circ]$ in einer Entfernung von mindestens $d_0 = 750\,\mathrm{m}$ erkannt werden. Eine Ausnahme stellt das Intervall $\epsilon = [260\,^\circ; 265\,^\circ]$ dar. Das fremde Luftfahrzeug muss hier bereits in einer Entfernung von $d_0 = 1000\,\mathrm{m}$ bekannt sein. Auch aus der Richtung $\epsilon = [320\,^\circ; 75\,^\circ]$ kommendes Luftfahrzeug muss zur Vermeidung einer *NMAC* in einer Entfernung von $d_0 = 1000\,\mathrm{m}$ erkannt werden.

4.3. Statische und bewegte Hindernisse

Das Kapitel wird mit einem Szenario aus einer Kombination der bereits vorgestellten Szenarien abgeschlossen. Das Ownship wird mittig in einem Geofence platziert. Der Geofence ist identisch mit dem in 4.1.3 vorgestellten Geofence. Zusätzlich werden zehn weitere Luftfahrzeuge im Geofence platziert. Sowohl die Startposition als auch die (konstante) Geschwindigkeit wird per Zufallsgenerator definiert. Die einzige Vorgabe ist, dass jedes Luftfahrzeug innerhalb des Geofences mit einer Fluggeschwindigkeit zwischen $v_{A,traffic[i]} = 15\,\mathrm{m/s}$ und $v_{A,traffic[i]} = 60\,\mathrm{m/s}$ startet. Die Flughöhe jedes Luftfahrzeugs ist dabei identisch mit der des Ownships ($z_g = -100\,\mathrm{m}$), eine Vertikalgeschwindigkeit ist für keines der Luftfahrzeuge erlaubt ($w = 0$). Abbildung 4.34 bietet eine schematische Darstellung des Szenarios.

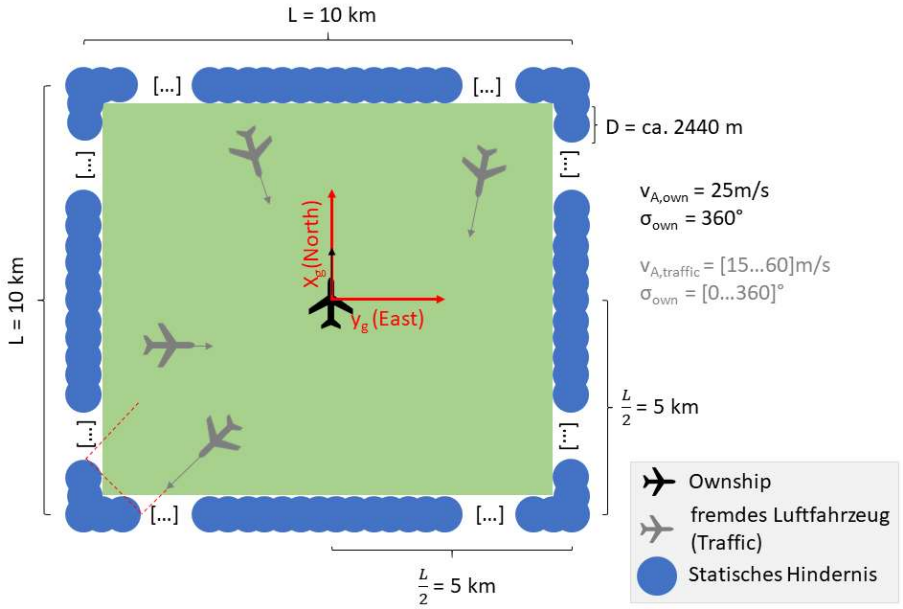


Abbildung 4.34 Schematische Darstellung Szenario statische und bewegte Hindernisse

Wie in obiger Abbildung zu sehen startet das Ownship im Koordinatenursprung mit einem Kurs von $\chi_{own} = 360\,^\circ$ und einer Geschwindigkeit von $v_{A,own} = 25\,\mathrm{m/s}$. Um einen möglichst intensiven Stresstest zu bieten, verlassen die Traffic Luftfahrzeuge den Geofence zu keinem Zeitpunkt. Sollten sie - wie in Abbildung 4.34 unten links angedeutet - an die Grenzen des Geofences stoßen, wird die jeweilige Geschwindigkeitskomponente invertiert und das Luftfahrzeug damit innerhalb des Geofence gehalten. Das Szenario wird für eine Dauer von $1000\,\mathrm{s}$[11] simuliert. Abbildung 4.35 zeigt die über die gesamte Flugdauer geplottete Distanzen zu den einzelnen bewegten Hindernissen.

[11] 16:40 min

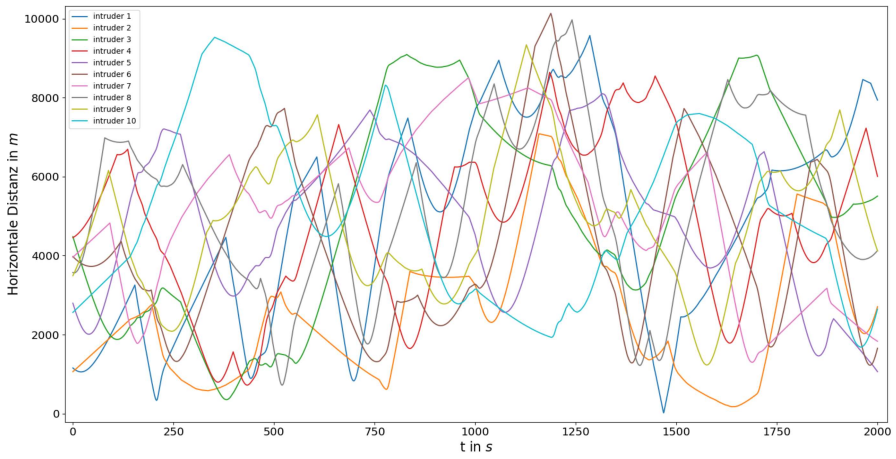


Abbildung 4.35 Distanzen zwischen Ownship und den bewegten Hindernissen

Sowohl das Ownship als auch die bewegten Hindernisse besitzen einen konstanten Geschwindigkeitsbetrag. Daraus kann gefolgert werden, dass die einzelnen Stellen mit linearer Distanzänderung für Teilstücke des Fluges stehen, in denen das Ownship kein Ausweichmanöver fliegt. Spitze Ecken in den Kurven sind als Umkehrzeitpunkte des jeweiligen Intruders am Rande des Geofences zu interpretieren. Es ist zu erkennen, dass die $1220\,\mathrm{m}$-Schwelle mehrere Male unterschritten wird. Es kommt zu insgesamt 20 Verletzungen der *Separation Volumes* der Intruder. Daneben kommt es zu vier *NMAC*, die geringste Distanz zu einem der Intruder beträgt bei $t = 1468\,\mathrm{s}$ lediglich $d_{min} = 19,4\,\mathrm{m}$. Der nächstgrößere Wert tritt mit $d_{min,2} = 174,6\,\mathrm{m}$ nach $t = 1638,5\,\mathrm{s}$ auf. Dagegen hält das Ownship die vorgesehene Distanz zu allen statischen Hindernissen, die den Geofence bilden, ein. Zu einer Kollision mit einem *Separation Volume* der statischen Hindernisse kommt es nicht. Je nachdem, um welche Art von Hindernis es sich bei den statischen Hindernissen in der Realität handelt, wäre eine Verletzung der *Separation Volumes* der statischen Hindernisse um im Gegenzug eine *NMAC* mit einem bewegten Intruder zu verhindern vorzuziehen. Eine Beurteilung hängt jedoch vom Einzelfall ab: Ein kurzer Einflug in das Randgebiet eines Naturschutzgebietes wäre weniger Sicherheitskritisch als das Verlassen eines durch Bebauung begrenzten Bereichs.
Aus Gründen der Vollständigkeit zeigt Abbildung 4.36 den vollständigen Flugpfad des Ownships innerhalb des Geofences mit den Ausweichmanövern gegenüber den Intrudern. Für eine übersichtliche Darstellung wird auf die Darstellung der Flugpfade der Intruder verzichtet.

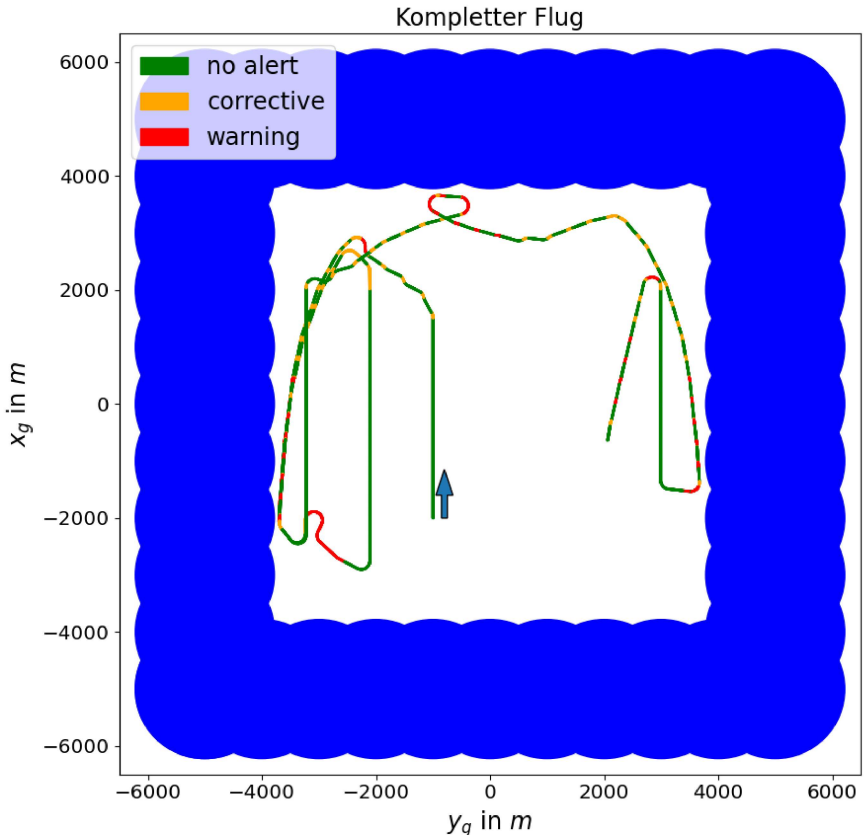


Abbildung 4.36 Flugpfad Szenario statische und bewegte Hindernisse

5. Zusammenfassung

Ziel dieser Arbeit ist eine Bewertung, inwiefern die von der NASA entwickelte DAIDALUS Programmbibliothek auch für DAA Szenarien von tieffliegenden UA genutzt werden kann. Dazu werden mehrere Szenarien simuliert und ausgewertet. Sowohl statische als auch bewegte Hindernisse sowie der kombinierte Fall wird betrachtet.
Kollisionen mit statischen Hindernissen werden mit der in dieser Arbeit gewählten Implementierung zuverlässig vermieden. Bei einer Anfangsdistanz von mindestens $d = 2000\,\mathrm{m}$ wird eine *NMAC* in jedem Fall vermieden. In einzelnen Fällen kann es zu leichten Verletzungen des *Recovery Volumes* kommen. Das Ownship nähert sich dem statischen Hindernis dabei auf bis zu $1053,2\,\mathrm{m}$ an. Bei einer Anfangsdistanz $d > 3000\,\mathrm{m}$ wird kurz nach Erreichen der 3000 m-Grenze das Ausweichmanöver gestartet. Zuvor fliegt das Ownship entlang des gewünschten Kurses. Die Ausrichtung und Position relativ zum statischen Hindernis hat ab einer Anfangsdistanz $d > 3000\,\mathrm{m}$ keinen Einfluss auf den Erfolg des Ausweichmanövers.
Eine Kombination von zwei statischen Hindernissen wird (wie in Abschnitt 4.1.2 gezeigt) ebenfalls erfolgreich umflogen. Das Ownship passiert die beiden statischen Hindernisse nur mittig, falls der Abstand zwischen den *Separation Volumes* der beiden Hindernisse bei einem Kurs des Ownships von $\chi = 45\,^\circ$ mindestens $\Delta y = 6600\,\mathrm{m}$ beträgt. Ist der Abstand der beiden statischen Hindernisse zueinander geringer, so werden beide Hindernisse umflogen. Bei einem Kurs des Ownships von $\chi = 360\,^\circ$ werden beide Hindernisse bereits ab einer Distanz von $\Delta h = 3000\,\mathrm{m}$ zueinander mittig durchflogen.
Ein Geofence kann aus vielen einzelnen statischen Hindernissen aufgebaut werden und sorgt nach den Versuchen in Abschnitt 4.1.3 dafür, dass das Ownship diesen nicht verlässt. Der im Versuch gewählte Geofence erweist sich als ausreichend groß gewählt und bietet dem Ownship genügend Raum für Wendemanöver. Die statischen Hindernisse, die die Begrenzung des Geofences bilden, überlappen sich in ausreichendem Maße, um ein Ausfliegen des Ownships zu verhindern.
Mit den in den DO-365A vorgegebenen Parametern für den DAA Algorithmus kann das Ownship dem in Abschnitt 4.1.4 definierten Flugkorridor nicht zuverlässig folgen. Stattdessen wird das *Recovery Volume* der jeweils an den inneren Ecken liegenden statischen Hindernisse verletzt. Das Ownship nähert sich dem statischen Hindernis dabei auf bis zu $1107,5\,\mathrm{m}$ an. Außerdem wendet das Ownship beim Abfliegen des Korridors mehrmals und fliegt in die entgegengesetzte Richtung. Wird die Lookahead Time von den in den DO-365A definierten $180\,\mathrm{s}$ auf $20\,\mathrm{s}$ reduziert, so wird dem in Abschnitt 4.1.4 verwendeten Flugkorridor zuverlässig gefolgt. Die *Separation Volumes* der statischen Hindernisse werden dabei nicht verletzt.
Auch bewegten Hindernissen wie anderen Luftfahrzeugen kann mit der gewählten Implementierung von DAIDALUS erfolgreich ausgewichen werden. Dabei reicht eine Distanz von $1000\,\mathrm{m}$ bei Erfassung des frontal entgegenkommenden bewegten Hindernisses (mit der in Abschnitt 4.2.1 gewählter Fluggeschwindigkeit) aus, um eine *NMAC* zu vermeiden. Die Verletzung der *Separation Volumes* wird ab einer Distanz von $2000\,\mathrm{m}$ verhindert. In Abschnitt 4.2.2 nähert sich ein Intruder von oben auf das Ownship in vertikaler Richtung an. In diesem

Szenario wird dem Intruder ab einer Höhendifferenz von mindestens $26\,\mathrm{m}$ zu Beginn des Fluges erfolgreich ausgewichen und eine *NMAC* verhindert.
Abhängig von der Fluggeschwindigkeit des Intruders und der Richtung, aus der der Intruder auf das Ownship zufliegt, ist eine frühere oder spätere Erfassung des Intruders nötig um eine *NMAC* oder die Verletzung der *Separation Volumes* zu verhindern. In Abschnitt 4.2.3 hat sich gezeigt, dass sich die nötige Distanz für von vorne beziehungsweise von hinten kommenden Intrudern zum Teil erheblich von der nötigen Distanz gegebenüber seitlich anfliegenden Intrudern unterscheidet. Fliegt der Intruder schneller als das Ownship, so müssen von vorne und von hinten kommende Intruder zum Teil deutlich früher erkannt werden als seitlich kommende Intruder. Ist der Intruder langsamer als das Ownship, so gleicht sich die nötige Distanz für von hinten kommenden Intrudern den von der Seite kommenden an.
Kombiniert man statische und bewegte Hindernisse in einem Szenario, so wird den statischen Hindernissen - wie in Abschnitt 4.3 gezeigt - zuverlässig ausgewichen und deren *Separation Volumes* nicht verletzt. Hingegen kann es zu *NMACs* mit den bewegten Hindernissen kommen. Zu beachten ist dabei, dass dieses Szenario das einzige in dieser Arbeit ist, in dem mehr als ein bewegtes Hindernis betrachtet wird.
Aus den präsentierten Ergebnissen lässt sich schließen, dass die im Rahmen dieser Arbeit erarbeitete Implementierung von DAIDALUS statischen und einzelnen bewegten Hindernissen jeweils zuverlässig ausweicht. Die dafür nötige Distanz bei der ersten Übergabe der Daten an DAIDALUS fällt je nach Anwendungsfall unterschiedlich aus. Hält sich das Ownship mit mehreren Luftfahrzeugen in einem begrenzten Gebiet auf, kann es hingegen zu *NMACs* kommen.

6. Ausblick

Die Einhaltung eines einfachen Geofences wurde in dieser Arbeit erfolgreich demonstriert. Im späteren Realbetrieb eines UAs sind komplexere Umrisse eines Geofences denkbar. Entsprechend muss das Verhalten innerhalb von komplexeren Geofences genauer analysiert werden. Weiterhin hat diese Arbeit einen Ansatz zur Erstellung eines Flugkorridors aus statischen Hindernissen getestet. Auch hier sind komplexere Strukturen für eine mögliche Flugführung denkbar. Um eine zuverlässige Einhaltung des gewünschten Flugkorridors zu gewährleisten, muss außerdem der Einfluss weiterer Parameter auf das Verhalten des Ownships im Flugkorridor erforscht werden. Hierbei sind vor allem die Lookahead Time in der Konfiguration von DAIDALUS sowie die räumliche Ausprägung des Flugkorridors in der Breite und der Kurvenradius des Flugkorridors zu beachten.
Darüber hinaus muss das Verhalten gegenüber mehreren bewegten Hindernissen weiter erforscht werden. Die Wahrscheinlichkeit, eine *NMAC* zu vermeiden sinkt mit der steigenden Dichte der Intruder pro Fläche beziehungsweise Volumen. Ein Zusammenhang zwischen der Dichte der Intruder pro Fläche und der Wahrscheinlichkeit zur Vermeidung einer *NMAC* ist wünschenswert.
Denkbar ist zudem die Verwendung von unterschiedlichen Alertern für unterschiedliche Hindernisarten. Beispielsweise geht von einer Kollision mit einem kleineren UA der Klasse *offen* eine geringere Gefahr für Personen- und/oder Sachschäden aus als von einer Kollision mit einem Rettungshubschrauber. Auch zur exakten Nachbildung von großen statischen Hindernissen wie Gebäuden können *Separation Volumes* mit unterschiedlichen Größen verwendet werden. Zudem kann es sinnvoll sein, eine gewisse Kategorisierung der bekannten Hindernisse vorzunehmen. Dadurch kann die Verletzung eines *Separation Volumes* eines nachgebildeten Luftraums in Kauf genommen werden, wenn stattdessen eine Kollision mit einem (bewohnten) Gebäude vermieden wird. Eine ähnliche Herangehensweise ist gegenüber den unterschiedlichen Arten von bewegten Hindernissen denkbar.
DAIDALUS ist Bestandteil der in Abschnitt 2.2.5 vorgestellten ICAROUS Softwarearchitektur. ICAROUS beinhaltet außerdem die PolyCARP Algorithmen. Durch diese Kombination scheint ICAROUS eine vielversprechende Lösung für ein DAA System gegenüber bewegten Hindernissen und statischen Hindernissen wie Geofences zu sein. Eine genauere Untersuchung der ICAROUS Softwarearchitektur bietet sich an.
Die Komplexität eines Ausweichmanövers kann um zusätzliche Dimensionen erweitert werden. Ausweichmanöver durch eine Änderung der Vertikal- und/oder Horizontalgeschwindigkeit bieten zusätzliche Freiheiten und Möglichkeiten gegenüber einem zweidimensionalen Ausweichmanöver durch eine Kursänderung. Dabei ist die geringe Flughöhe des ALAADy-Demonstrators im vorgesehenen Einsatzzweck zu beachten. Um eine Kollision mit Geländererhöhungen oder dem Boden zu verhindern muss der ALAADy-Demonstrator durch den Einsatz von Sensoren wie zum Beispiel LIDAR oder SONAR oder durch den Einsatz von Geländedatenbanken in der Lage sein, einen Absturz zu verhindern. Außerdem muss für den Mischbetrieb mit unbemannten und bemannten Luftfahrzeugen in einem Luftraum dafür ge-

sorgt werden, dass bei der Berechnung der Ausweichmanöver die Ausweichregeln der Rules of the Air [41] eingehalten werden.
In jedem Fall muss die später im Realbetrieb zu verwendende Software vorab ausgiebig in der Simulation des dynamischen Bewegungsmodells getestet werden. Darüber hinaus muss die Echtzeitfähigkeit des Systems durch eine Hardware-in-the-Loop Simulation nachgewiesen werden.
Für eine umfassende Beobachtung der realen Testflüge muss außerdem eine Möglichkeit gegeben sein, die Daten des ALAADy-Demonstrators in Echtzeit am Boden mitverfolgen zu können. Neben den üblichen Daten zu Fluggeschwindigkeit, Kurs, Flughöhe und ähnlichem müssen bei der Erprobung des DAA Systems die zugehörigen Daten übersichtlich dargestellt werden. Dazu gehören beispielsweise die Position und die Fluggeschwindigkeit und -richtung der bekannten Intruder und statischen Hindernissen. DAIDALUS bietet die Ausgabe von sogenannten Bändern, innerhalb derer eine Kollisionsgefahr besteht. Eine derartige Visualisierung kann sich beispielsweise am DANTi Konzept der NASA [14] orientieren.

Literatur

[1] The executive departments & agencies of the federal government of the United States. *14 CFR 121.356 - Code of Federal Regulations*. Legal Rule or Regulation. 2003.

[2] The executive departments & agencies of the federal government of the United States. *14 CFR 91.119(c) - Code of Federal Regulations*. Legal Rule or Regulation. 2010.

[3] FLARM Technology AG. *About Us -History*. Web Page. URL: https://flarm.com/about-us/history/,12.09.2022.

[4] FLARM Technology AG. *FLARM Data Port Specification*. Standard. 2015.

[5] FLARM Technology AG. *FLARM Hindernisdatenbank*. Web Page. 2022. URL: https://flarm.com/de/produkt-kategorie/hindernisdatenbanken/,21.08.2022.

[6] FLARM Technology AG. *FLARM Solutions*. Web Page. URL: https://www.flarm.com/solutions/for-pilots-aircraft-owners/solutions-for-general-aviation/,31.01.2023.

[7] FLARM Technology AG. *The Affordable Collision Avoidance Technology for General Aviation and UAV*. Electronic Article. Dez. 2020. URL: https://www.flarm.com/wp-content/uploads/man/FLARM-General-EN.pdf,20.12.2022.

[8] European Defence Agency. *MIDCAS demonstrates progress for RPAS integration into civil airspace*. Electronic Article. 2015.

[9] Unmanned Airspace. *European Union to provide EUR21 million for military RPAS detect and avoid research*. Electronic Article. 2020.

[10] Cyril Allignol, Nicolas Barnier, Nicolas Durand und Éric Blond. *Detect & Avoid, UAV Integration in the Lower Airspace Traffic*. Electronic Article. 2016.

[11] Swee Balachandran, Anthony Narkawicz, César Munoz, María Consiglio, NIA und NASA. *A Path Planning Algorithm to Enable Well-Clear Low Altitude UAS Operation Beyond Visual Line of Sight*. Conference Paper. 2017.

[12] Sebastian Benders, Lukas Goormann, Sven Lorenz, Johann Dauer und DLR. *Softwarearchitektur für einen unbemannten Luftfrachttransportdemonstrator*. Conference Paper. 2018.

[13] Sebastian Benders, Sven Lorenz und DLR. *Automated Ground Operation for an Unmanned Cargo Gyrocopter*. Electronic Article. 2022.

[14] Victor A. Carreno, Compass Engineering und NASA. *Evaluation, Analysis and Results of the DANTi Flight Test Data, the DAIDALUS Detect and Avoid Algorithm, and the DANTi Concept for Detect and Avoid in the Cockpit*. Electronic Article. 2020.

[15] Land Processes Distributed Active Archive Center. *NASA Shuttle Radar Topography Mission (SRTM) Global 1 arc second Data Released Over the Middle East*. Web Page. 2015. URL: https://lpdaac.usgs.gov/news/nasa-shuttle-radar-topography-mission-srtm-global-1-arc-second-data-released-over-the-middle-east/, 25.10.2022.

[16] Reece Clothier, Brendan Williams und Neale Fulton. *Structuring the safety case for unmanned aircraft system operations in non-segregated airspace*. Electronic Article. 2015. DOI: 10.1016/j.ssci.2015.06.007.

[17] Darren Cofer, Ramachandra Sattigeri, Isaac Amundson, Junaid Babar, Saqib Hasan, Eric W. Smith, Karthik Nukala, Denis Osipychev, Lucca Timmerman, Dragos D. Margineantu, James L. Paunicka und Matthew A. Moser. *Flight Test of a Collision Avoidance Neural Network with Run-Time Assurance*. Electronic Article. Jan. 2022.

[18] European Commission. *EU 2020/587*. Legal Rule or Regulation. 2020.

[19] European Commission. *Regulation EU 2019/947*. Legal Rule or Regulation. 2019.

[20] ArduPilot Community. *ArduPilot Documentation - Fences*. Web Page. 2022. URL: https://ardupilot.org/copter/docs/common-geofencing-landing-page.html, 21.10.2022.

[21] ArduPilot Community. *ArduPilot Documentation - Terrain Following*. Web Page. 2022. URL: https://ardupilot.org/copter/docs/terrain-following.html, 21.10.2022.

[22] Ardupilot Community. *ArduPilot Documentation*. Web Page. 2022. URL: https://ardupilot.org/ardupilot/, 21.10.2022.

[23] Ardupilot Community. *ArduPilot Documentation - ADS-B*. Web Page. 2022. URL: https://ardupilot.org/copter/docs/common-ads-b-receiver.html?highlight=avoidance, 21.10.2022.

[24] PX4 Autopilot Community. *PX4 Autopilot User Guide - Air Traffic Avoidance*. Web Page. 2022.

[25] PX4 Autopilot Community. *PX4 Autopilot User Guide - main*. Web Page. 2022. URL: https://docs.px4.io/main/en/, 21.10.2022.

[26] PX4 Autopilot Community. *PX4 User Guide - GeoFence*. Web Page. 2022. URL: https://docs.px4.io/main/en/flying/geofence.html, 21.12.2022.

[27] Jason T. Davies, Minghong G. Wu, Universities Space Research Association und Ames Research Center. *Comparative Analysis of ACAS-Xu and DAIDALUS Detect-and-Avoid Systems*. Electronic Article. 2018.

[28] Adrian Dudek, Peter Stütz, Florian Kunstmann und Jens Hennig. *Detect and Avoid of Weather Phenomena on-board UAV: Increasing Detection Capabilities by Information Fusion*. Electronic Article. 2021. DOI: 978-1-6654-3420-1/21/$31.00.

[29] Aaron Dutle. *E-Mailverkehr mit Mitentwickler von DAIDALUS*. Personal Communication. 2022.

[30] Deutscher Aero Club e.V. *DAEC - Luftraumdaten Deutschland*. Online Database. 2022. URL: www.daec.de/fachbereiche/luftraum-flugsicherheit-flugbetrieb/luftraumdaten/, 20.08.2022.

[31] Deutscher Aero Club e.V. *Transponder Mode S*. Electronic Article. 2002. URL: daec.de/fileadmin/user_upload/files/2012/fachbereiche/luftfahrttechnik/transponder_mode_s.pdf, 20.08.2022.

[32] Stefan Erhardt. *OpenTopoMap - About*. Web Page. 2022. URL: https://opentopomap.org/about, 25.10.2022.

[33] EUROCAE. *ED-267 Operational Services & Environment Definition (OSED) for Detect & Avoid in very low-level Operations*. Legal Rule or Regulation. 2020.

[34] EUROCONTROL. *NETALERT Newsletter No17*. Electronic Article. 2013.

[35] FAA. *Introduction to TCAS II Version 7.1*. Generic. 2011.

[36] Deutsche Flugsicherung. *ATP-Online*. Online Database. 2022. URL: https://ais.dfs.de/pilotservice/service/information/aip_online/aip_online.jsp, 20.08.2022.

[37] Deutsche Flugsicherung. *LIZ Annual Summary 2021*. Electronic Article. 2022.

[38] OpenStreetMap Foundation. *OpenStreetMap - Karte*. Online Database. 2022. URL: https://www.openstreetmap.org/, 25.10.2022.

[39] OpenStreetMap Foundation. *OpenStreetMap FAQ - Wie vollständig sind die Daten?* Web Page. 2022. URL: https://openstreetmap.de/faq/#wie-vollst%C3%A4ndig-sind-die-daten, 25.10.2022.

[40] IATA und Honeywell. *Guidance material - Performance assessment of pilot response to Enhanced Ground Proximity Warning System (EGPWS)*. Electronic Article. 2019.

[41] ICAO. *Annex 2 - Rules of the Air*. Legal Rule or Regulation. 2005.

[42] RTCA Inc. *DO-365: Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems*. Legal Rule or Regulation. 2017.

[43] RTCA Inc. *DO-365A: Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems*. Legal Rule or Regulation. 2020.

[44] Bundesamt für Kartographie und Geodäsie. *Lufträume der Deutschen Flugsicherung*. Web Page. 2022. URL: https://gdz.bkg.bund.de/index.php/default/luftraeume-der-dfs-luftraumdfs.html, 20.08.2022.

[45] Sydney M. Katz, Mykel J. Kochenderfer, Luis E. Alvarez, Michael Owen, Samuel Wu, Marc Brittain und Anshuman Das. *Collision Risk and Operational Impact of Speed Change Advisories as Aircraft Collision Avoidance Maneuvers*. Electronic Article. 2022.

[46] Zhengrong Li, Yuee Liu, Ross Hayward, Jinglan Zhang und Jinhai Cai. *Knowledge-based Power Line Detection for UAV Surveillance and Inspection Systems*. Electronic Article. 2008. DOI: 978-1-4244-2582-2/08.

[47] Xiong Liu, Lin Hou und Xiongming Ju. *A Method For Detecting Power Lines In UAV Aerial Images*. Electronic Article. 2017. DOI: 978-1-5090-6352-9/17.

[48] Deutsches Zentrum für Luft- und Raumfahrt e.V. *ALAADy Foto*. Figure. 2019. URL: https://www.dlr.de/ft/Portaldata/18/Resources/images/projekte/alaady-cc-update_2022/Alaady_2019_07_04_25_1440x960.jpg, 12.01.2023.

[49] Deutsches Zentrum für Luft- und Raumfahrt e.V. *Die Welt in 3D zum kostenlosen Download*. Blog. 2018. URL: https://www.dlr.de/eoc/desktopdefault.aspx/tabid-12632/22039_read-53088/, 25.10.2022.

[50] Deutsches Zentrum für Luft- und Raumfahrt e.V. *Internetauftritt ALAADy*. Web Page. URL: https://www.dlr.de/ft/alaady, 12.01.2023.

[51] Luftfahrtbundesamt. *Onlinekurs - UAS Klassifizierung*. Web Page. 2022. URL: https://lba-openuav.de/onlinekurs/lehrmaterial/luftrecht-und-sicherheit/uas-klassifizierungen/, 13.08.2022.

[52] Lorenz Meier. *PX4 Autopilot User Guide - GeoFence*. Web Page. 2022. URL: https://docs.px4.io/main/en/flying/geofence.html, 22.10.2022.

[53] Lorenz Meier. *PX4 Autopilot User Guide - Terrain Following*. Web Page. 2022. URL: https://docs.px4.io/main/en/flying/terrain_following_holding.html, 22.10.2022.

[54] G. Migliaccio, G. Mengali, R. Galatolo und University of Pisa. *Conflict detection and resolution algorithms for UAVs collision avoidance*. Electronic Article. 2014.

[55] César Munoz und Anthony Narkawicz. *Formal Analysis of Extended Well-Clear Boundaries for Unmanned Aircraft*. Electronic Article. 2016.

[56] César Munoz, Anthony Narkawicz, George Hagen, Jason Upchurch, Aaron Dudle, María Consiglio, NASA, James Chamberlain und Sunrise Aviation Inc. *DAIDALUS: Detect and Avoid Alerting Logic for Unmanned Systems*. Conference Paper. 2015.

[57] Anthony Narkawicz, César Munoz, Aaron Dutle und NASA. *Sensor Uncertainty Mitigation and Dynamic Well Clear Volumes in DAIDALUS*. Generic.

[58] NASA. *DAIDALUS Github Repository*. Web Page. 2022. URL: https://github.com/nasa/daidalus, 10.08.2022-10.02.2023.

[59] NASA. *ICAROUS*. Computer Program. 2022. URL: https://github.com/nasa/icarous, 15.08.2022.

[60] NASA. *PolyCARP*. Computer Program. 2017. URL: https://github.com/nasa/PolyCARP, 15.08.2022.

[61] NASA. *Reference Manual - DAIDALUS-v2.0.x*. Web Page. 2022. URL: https://nasa.github.io/daidalus/, 21.12.2022.

[62] Michael P. Owen, Adam Panken, Robert Moss, Luis Alvarez und Charles Leeper. *ACAS Xu: Integrated Collision Avoidance and Detect and Avoid Capability for UAS*. Conference Paper. 2019. DOI: 10.1109/DASC43569.2019.9081758.

[63] The Paparazzi Project. *Paparazzi UAV Wiki*. Web Page. URL: https://wiki.paparazziuav.org/wiki/Main_Page, 04.02.2023.

[64] Eric Schafer und Iris Automation. *Sense and Avoid: How it Works in Unmanned Aerial Vehicles*. Blog. 2021. URL: https://www.irisonboard.com/how-sense-and-avoid-works-in-unmanned-aerial-vehicles/, 13.08.21.

[65] Ren Tianzhu, Zhou Rui, Xia Jie und Dong Zhuoning. *Three-dimensional Path Planning of UAV Based On an Improved A* Algorithm*. Electronic Article. 2016. DOI: 10.1109/CGNCC.2016.7828772.

[66] Bundesministerium für Verkehr und digitale Infrastruktur. *Unbemannte Luftfahrtsysteme und innovative Luftfahrtkonzepte Aktionsplan der Bundesregierung*. Legal Rule or Regulation. 2020.

[67] Deutscher Wetterdienst. *DWD-Geoportal*. Web Page. 2022. URL: https://dwd-geoportal.de/, 18.09.2022.

[68] Deutscher Wetterdienst. *OpenData DWD - LIESMICH*. Web Page. 2022. URL: https://opendata.dwd.de/LIESMICH.txt, 18.09.2022.

[69] Deutscher Wetterdienst. *Wetter- und Klimalexikon: Mesozyklone*. Web Page. URL: https://www.dwd.de/DE/service/lexikon/Functions/glossar.html?lv3=605644&lv2=101640, 31.08.2022.

Anhang

A. DAIDALUS Konfigurationsdatei

Die hier hinterlegte Konfigurationsdatei entspricht der in der Arbeit verwendeten. Sie basiert auf der im GitHub Repository von DAIDALUS [58] bereitgestellten Konfigurationsdatei DO_-365A_SUM.conf. Diese Datei wurde um die Performancegrenzen des ALAADy-Demonstrators ergänzt und um einen optionalen Wert für die `lookahead_time` ergänzt.

```
## based on the DO_365A_SUM.conf provided on Github, this file is
## an attempt to get a proper configuration for the
## ALAADy-Demonstrator

## values estimated for autopilot-cruisespeed

# Daidalus Object
# V-2.0.2c
# Bands Parameters
lookahead_time = 180.0 [s]
#lookahead_time = 20.0 [s]
left_hdir = 180.0 [deg]
right_hdir = 180.0 [deg]
min_hs = 40.0 [kph]
max_hs = 145.0 [kph]
min_vs = -7.0 [m/s]
max_vs = 5.0 [m/s]
min_alt = 50.0 [m]
max_alt = 120.0 [m]
# Relative Bands Parameters
below_relative_hs = 0.0 [kph]
above_relative_hs = 0.0 [kph]
below_relative_vs = 0.0 [m/s]
above_relative_vs = 0.0 [m/s]
below_relative_alt = 0.0 [m]
above_relative_alt = 0.0 [m]
# Kinematic Parameters
step_hdir = 1.0 [deg]
```

```
29 step_hs = 5.0 [kph]
30 step_vs = 0.5 [m/s]
31 step_alt = 5.0 [m]
32 horizontal_accel = 2.0 [m/s^2]
33 vertical_accel = 0.2 [G]
34 turn_rate = 0.1666 [rad/s]
35 bank_angle = 0 [deg]#23.0 [deg]
36 vertical_rate = 3.0 [m/s] #500.0 [fpm]
37 # Recovery Bands Parameters
38 min_horizontal_recovery = 500.0 [m]
39 min_vertical_recovery = 45.0 [m]
40 recovery_hdir = true
41 recovery_hs = false #true
42 recovery_vs = false #true
43 recovery_alt = false #true
44 # Collision Avoidance Bands Parameters
45 ca_bands = true
46 ca_factor = 0.1
47 horizontal_nmac = 500.0 [m]
48 vertical_nmac = 20.0 [m]
49 # Hysteresis and persistence parameters
50 recovery_stability_time = 3.0 [s]
51 hysteresis_time = 5.0 [s]
52 persistence_time = 4.0 [s]
53 bands_persistence = true
54 persistence_preferred_hdir = 5.0 [deg] #15.0 [deg]
55 persistence_preferred_hs = 25.0 [knot] #100.0 [knot]
56 persistence_preferred_vs = 250.0 [fpm]
57 persistence_preferred_alt = 250.0 [ft]
58 alerting_m = 2
59 alerting_n = 4
60 # Implicit Coordination Parameters
61 conflict_crit = false
62 recovery_crit = false
63 # Sensor Uncertainty Mitigation Parameters
64 h_pos_z_score = 1.5
65 h_vel_z_score_min = 0.5
66 h_vel_z_score_max = 1.0
67 h_vel_z_distance = 5.0 [nmi]
68 v_pos_z_score = 0.75
69 v_vel_z_score = 1.5
```

```
70  # Horizontal Contour Threshold
71  contour_thr = 180.0 [deg]
72  # DAA Terminal Area (DTA)
73  dta_logic = 0
74  dta_latitude = 0.0 [deg]
75  dta_longitude = 0.0 [deg]
76  dta_radius = 4.2 [nmi]
77  dta_height = 2000.0 [ft]
78  dta_alerter = 2
79
80  # MY Custom Alerter (MCA)
81  mca_logic = 0
82  mca_alerter = 3
83
84  # Alerting Logic
85  ownship_centric_alerting = false
86  corrective_region = MID
87  alerters = DWC_Phase_I_SUM,DWC_Phase_II_SUM
88  DWC_Phase_I_SUM_alert_1_region = NONE
89  DWC_Phase_I_SUM_alert_1_alerting_time = 50.0 [s]
90  DWC_Phase_I_SUM_alert_1_early_alerting_time = 75.0 [s]
91  DWC_Phase_I_SUM_alert_1_spread_hdir = 0.0 [deg]
92  DWC_Phase_I_SUM_alert_1_spread_hs = 0.0 [knot]
93  DWC_Phase_I_SUM_alert_1_spread_vs = 0.0 [fpm]
94  DWC_Phase_I_SUM_alert_1_spread_alt = 0.0 [ft]
95  DWC_Phase_I_SUM_alert_1_detector = det_1
96  DWC_Phase_I_SUM_det_1_WCV_DTHR = 0.66 [nmi]
97  DWC_Phase_I_SUM_det_1_WCV_ZTHR = 700.0 [ft]
98  DWC_Phase_I_SUM_det_1_WCV_TTHR = 35.0 [s]
99  DWC_Phase_I_SUM_det_1_WCV_TCOA = 0.0 [s]
100 DWC_Phase_I_SUM_load_core_detection_det_1 =
101     gov.nasa.larcfm.ACCoRD.WCV_TAUMOD_SUM
102 DWC_Phase_I_SUM_alert_2_region = MID
103 DWC_Phase_I_SUM_alert_2_alerting_time = 50.0 [s]
104 DWC_Phase_I_SUM_alert_2_early_alerting_time = 75.0 [s]
105 DWC_Phase_I_SUM_alert_2_spread_hdir = 0.0 [deg]
106 DWC_Phase_I_SUM_alert_2_spread_hs = 0.0 [knot]
107 DWC_Phase_I_SUM_alert_2_spread_vs = 0.0 [fpm]
108 DWC_Phase_I_SUM_alert_2_spread_alt = 0.0 [ft]
109 DWC_Phase_I_SUM_alert_2_detector = det_2
110 DWC_Phase_I_SUM_det_2_WCV_DTHR = 0.66 [nmi]
```

```
111 DWC_Phase_I_SUM_det_2_WCV_ZTHR = 450.0 [ft]
112 DWC_Phase_I_SUM_det_2_WCV_TTHR = 35.0 [s]
113 DWC_Phase_I_SUM_det_2_WCV_TCOA = 0.0 [s]
114 DWC_Phase_I_SUM_load_core_detection_det_2 =
115     gov.nasa.larcfm.ACCoRD.WCV_TAUMOD_SUM
116 DWC_Phase_I_SUM_alert_3_region = NEAR
117 DWC_Phase_I_SUM_alert_3_alerting_time = 25.0 [s]
118 DWC_Phase_I_SUM_alert_3_early_alerting_time = 55.0 [s]
119 DWC_Phase_I_SUM_alert_3_spread_hdir = 0.0 [deg]
120 DWC_Phase_I_SUM_alert_3_spread_hs = 0.0 [knot]
121 DWC_Phase_I_SUM_alert_3_spread_vs = 0.0 [fpm]
122 DWC_Phase_I_SUM_alert_3_spread_alt = 0.0 [ft]
123 DWC_Phase_I_SUM_alert_3_detector = det_3
124 DWC_Phase_I_SUM_det_3_WCV_DTHR = 0.66 [nmi]
125 DWC_Phase_I_SUM_det_3_WCV_ZTHR = 450.0 [ft]
126 DWC_Phase_I_SUM_det_3_WCV_TTHR = 35.0 [s]
127 DWC_Phase_I_SUM_det_3_WCV_TCOA = 0.0 [s]
128 DWC_Phase_I_SUM_load_core_detection_det_3 =
129     gov.nasa.larcfm.ACCoRD.WCV_TAUMOD_SUM
```

B. Abbildungen zu Abschnitt 4.2.3

Nachfolgend ist eine Ergänzung der in Abschnitt 4.2.3 gezeigten Abbildungen gegeben.

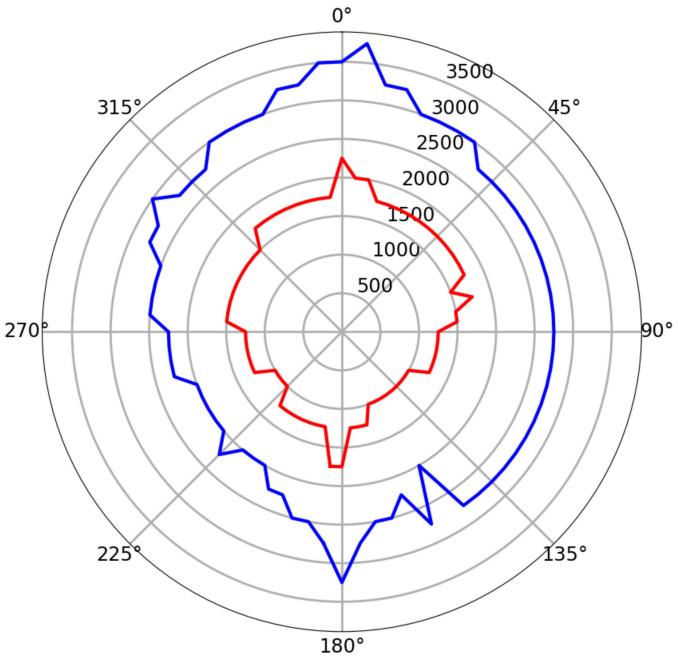


Abbildung B.1 Mindeststartdistanzen Fall (a): $v_{A,own} < v_{A,traffic} = 60\,\mathrm{m/s}$

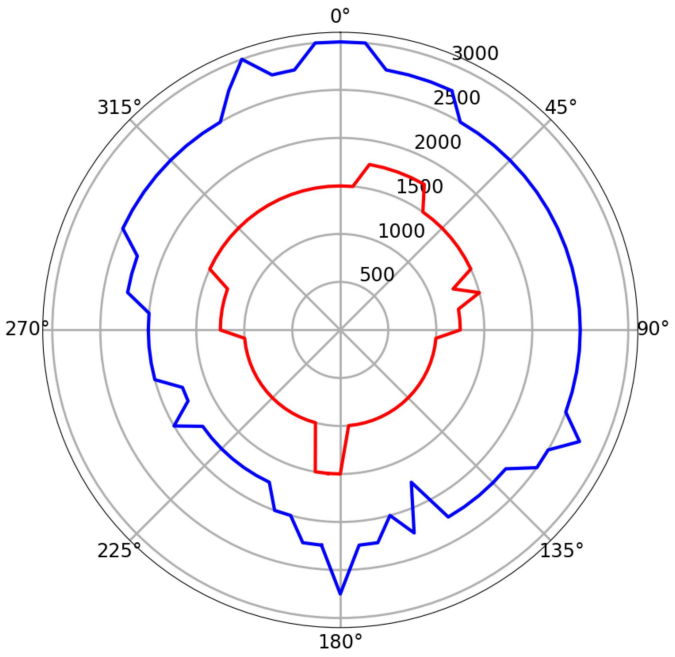


Abbildung B.2 Mindeststartdistanzen Fall (a): $v_{A,own} < v_{A,traffic} = 50\,\mathrm{m/s}$

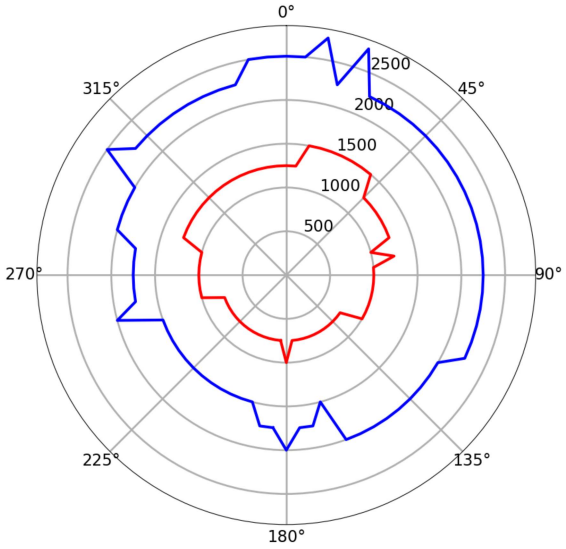


Abbildung B.3 Mindeststartdistanzen Fall (a): $v_{A,own} < v_{A,traffic} = 40\,\mathrm{m/s}$

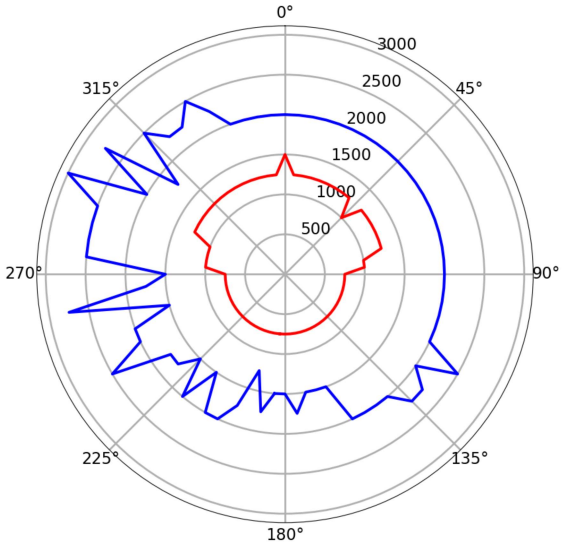


Abbildung B.4 Mindeststartdistanzen Fall (a): $v_{A,own} < v_{A,traffic} = 30\,\mathrm{m/s}$

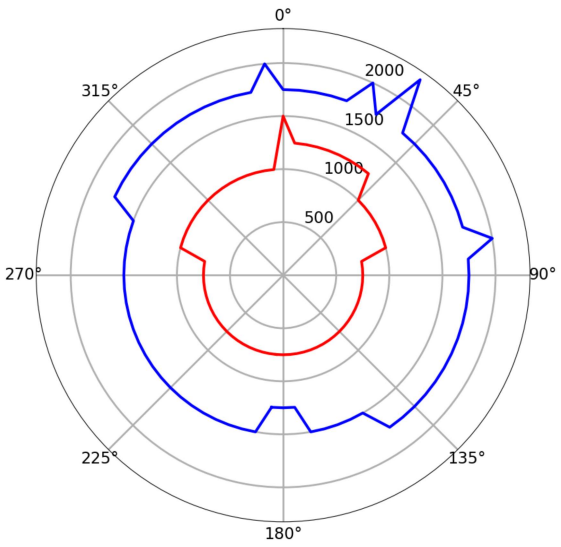


Abbildung B.5 Mindeststartdistanzen Fall (b): $v_{A,own} = v_{A,traffic} = 25\,\mathrm{m/s}$

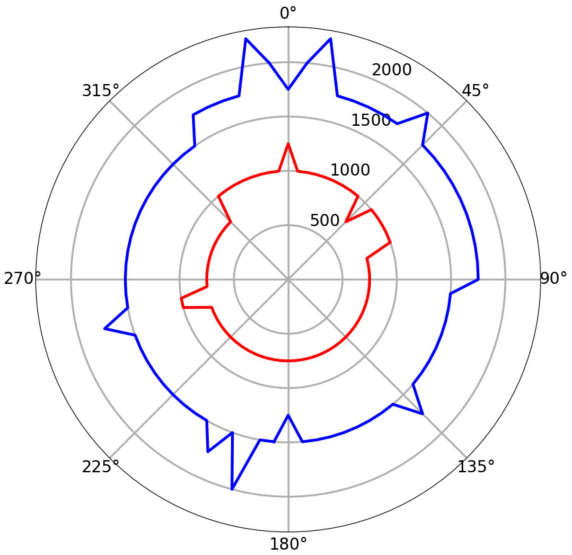


Abbildung B.6 Mindeststartdistanzen Fall (a): $v_{A,own} > v_{A,traffic} = 15\,\mathrm{m/s}$