Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Prof. Dr.-Ing. Hakan Kayal
Professur für Raumfahrttechnik
Lehrstuhl Informatik VIII

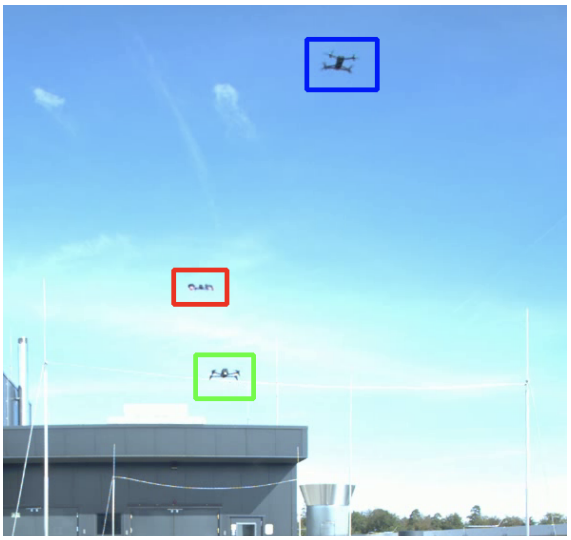
Masterarbeit

Drohnenerkennung mittels Bildverarbeitung durch eine 360° Kamera

von

Jonas Lutz

zur Erlangung des akademischen Grades
„Master of Science“
im Fach Informatik



30. Oktober 2018

betreut durch
Prof. Dr. Hakan KAYAL

Kurzfassung

Das Institut für Technische Physik am **D**eutsches Zentrum für **L**uft- und **R**aumfahrt (DLR) in Stuttgart befasst sich insbesondere mit der Entwicklung von neuartigen Laserquellen, deren Integration in optische Systeme und bewertet die Leistungsfähigkeit unter dem Einfluss der Atmosphäre, um zukünftig die Luft- und Raumfahrt sicherer zu gestalten. **U**nmanned **A**erial **V**ehicle (UAV) stellen zunehmend ein Problem für die Luftfahrt sowie die Sicherheit am Boden dar. Fehlverhalten der UAV oder sogar terroristische Aktivitäten führen zu neuen Gefährdungen, die rechtzeitig erkannt und mit geeigneten Mitteln abgewehrt werden müssen. Als erster Schritt ist dazu immer eine automatisierte Detektion der **U**nmanned **A**erial **S**ystem (UAS) Objekte im Luftraum erforderlich. Hierzu wurde im Rahmen der vorliegenden Masterarbeit eine passiv-optische Detektionseinheit für ein zukünftiges Drohnenabwehrsystem entwickelt. Als Sensor wird ein statisch feststehendes und kommerziell erhältliches 360 Grad Kamerasystem eingesetzt und hinsichtlich seiner optischen Eigenschaften (Auflösung, Kontrast, Bildrate, Stichingqualität, Winkelreferenzierung) bewertet. Ziel war es geeignete Bildverarbeitungsalgorithmen in der Entwicklungsumgebung Visual Studio 15 in C++ zu entwerfen, um ein oder mehrere fliegende UAS Objekte im Luftraum mit einer max. Entfernung von 30 Meter zu erfassen. In einer geeigneten Testumgebung im Außenbereich wurden die ausgewählten Bildverarbeitungsalgorithmen, die mittels der Bibliothek von **Open** Source **C**omputer **V**ision Library (OpenCV) verwirklicht wurden, in Kombination mit ihren Einstellparameter getestet und optimiert. Der Einfluss von wechselnden Licht- und Wetterbedingungen sowie die Falschalarmrate, wie z.B. durch Vögel, Personen und Fahrzeuge, wurden so weit wie möglich minimiert. Hierzu kommen sowohl räumliche als auch zeitliche Filterfunktionen zum Einsatz. Mittels Verfahren der Bildverarbeitung werden potentielle Objekte für den Anwender markiert (ROI) und mit einer Priorität gewichtet. Alle vom User-in-the-Loop erkannten und markierten UAS Objekte sollen mit ihrer referenzierten Winkelposition und Zeit geloggt und mittels geeigneter Datenübergabe an ein bereits bestehendes optisches Tracking System möglichst zeitnah und kontinuierlich übergeben werden. Dieses höher aufgelöste optische Tracking System ist auf einer 360° Schwenk- und Neigeplattform montiert und ermöglicht eine vom Detektionssystem unabhängige Nachverfolgung und Identifizierung des markierten UAS. Da die Bildverarbeitung des Detektionssystems zu Demonstrationszwecken unter variablen Umgebungsbedingungen eingesetzt werden soll, ist neben einer benutzerfreundlichen **G**raphical **U**nit **I**nterface (GUI) auch die Speicherung unterschiedlicher Settings der Bildverarbeitungsalgorithmen erforderlich. Abschließend wurde das UAS Detektionssystem auf seine Leistungsfähigkeit bewertet und vorgeführt. Die Ergebnisse daraus wurden kompakt in einem Datenblatt festgehalten.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Würzburg, 30. Oktober 2018

Danksagung

An dieser Stelle möchte ich mich bei Prof. Dr.-Ing. Kayal für die Betreuung der Arbeit bedanken. Weiterhin möchte ich mich bei Dr. Ivo Buske und Andreas Walther für die Vergabe des Themas und die Betreuung während der Arbeit beim Deutschen Zentrum für Luft- und Raumfahrt bedanken. Des Weiteren bedanke ich mich bei Daniel Fitz für den wertvollen Input und die konstruktive Kritik beim Schreiben dieser Arbeit. Abschließend gilt ein besonderer Dank meinen Eltern Michaela und Klaus Lutz, sowie Georg Winheim und Joachim Illmer für das Korrekturlesen der Arbeit und die Unterstützung während meines Studiums.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADC	**A**nalog-/ **D**igital **C**onverter
API	**A**pplication **P**rogramming **I**nterface
DLR	**D**eutsches Zentrum für **L**uft- und **R**aumfahrt
FOV	**F**ield **O**f **V**iew
GUI	**G**raphical **U**nit **I**nterface
HSV	**H**ue **S**aturation **V**alue
IOSB	Fraunhofer-**I**nstitut für **O**ptronik, **S**ystemtechnick und **B**ildauswertung
OpenCV	**Open** Source **C**omputer **V**ision Library
ROI	**R**egion **O**f **I**nterest
SDK	**S**oftware **D**evelopment **K**it
SUN	**S**chwenk- **u**nd **N**eigeplattform
TCP/IP	**T**ransmission **C**ontrol **P**rotocol /**I**nternet **P**rotocol
UAV	**U**nmanned **A**erial **V**ehicle
UAS	**U**nmanned **A**erial **S**ystem

1 Einleitung

Unmanned **A**erial **V**ehicles (UAVs) sind mittlerweile nicht mehr aus dem Alltag wegzudenken. So werden diese einerseits von den öffentlichen Behörden benutzt, um Menschenmassen zu überwachen, andererseits wird der kommerzielle Sektor für den gewerblichen und privaten Verkauf immer größer. Hier werden diese für verschiedenste Aufgaben genutzt, wie z.B. das Filmen von bestimmten Bereichen von oben, um einerseits in der Landwirtschaft bessere Analysen der Ertragsflächen von Felder machen zu können, andererseits ermöglichen UAVs Privatpersonen neue Videoperspektiven für den Privatgebrauch. Begünstigt durch eine sehr schnelle Entwicklung in den letzten Jahren, können heutzutage UAVs sehr günstig auf allen bekannten Onlineplattformen bereits für weniger als 50 € gekauft werden. Auch werden diese immer benutzerfreundlicher, da die Steuerung in den letzten Jahren durch neue Regelungstechnikalgorithmen enorm verbessert wurde. Weiterhin wird die Benutzerfreundlichkeit durch die immer anwendungsfreundlicheren Features unterstützt. Für die Steuerung wird lediglich ein Smartphone und eine darauf installierte App, die in den App-Stores aller gängigen Smartphonehersteller verfügbar ist, benötigt und schon kann die Drohne inklusive Live-Sicht über die integrierte Kamera sehr einfach via 2,4 GHz W-LAN geflogen werden. Oft stellen diese Apps, je nach Preiskategorie der Drohne, auch Funktionen für eine automatisierte Flugrouteneingabe via GPS bereit, wodurch ein autonomes Abfliegen der eingegebenen Punkte mögich ist. Oben genannte Punkte haben zur Folge, dass steigende Verkaufszahlen in den nächsten Jahren prognostiziert werden, was folgendende Studie beweist: "Verbraucher werden laut einer aktuellen Studie des US-Marktforschungsunternehmens Tractica in diesem Jahr weltweit 10,2 Millionen Drohnen für den privaten Gebrauch kaufen. Bis 2021 soll der Absatz auf rund 68 Millionen steigen. Das entspricht einer Verzehnfachung der Nachfrage gegenüber dem Jahr 2015."[dro18] Die Entwicklung der Verkauszahlen ist in Abbildung 1.1 dargestellt.

Weiterhin schreitet die Entwicklung fort und es wurden erste Konzepte für die Schwarmintelligenz von Drohnen bereits vorgestellt (vergl. [sch18]). Auch werden in Zukunft die Geschwindigkeit und Flugzeiten sowie die Autonomität der Drohnen weiterhin steigen. Eine Übersicht der aktuellen Fähigkeiten von Drohnen wird am Beispiel DJI vorgestellt.

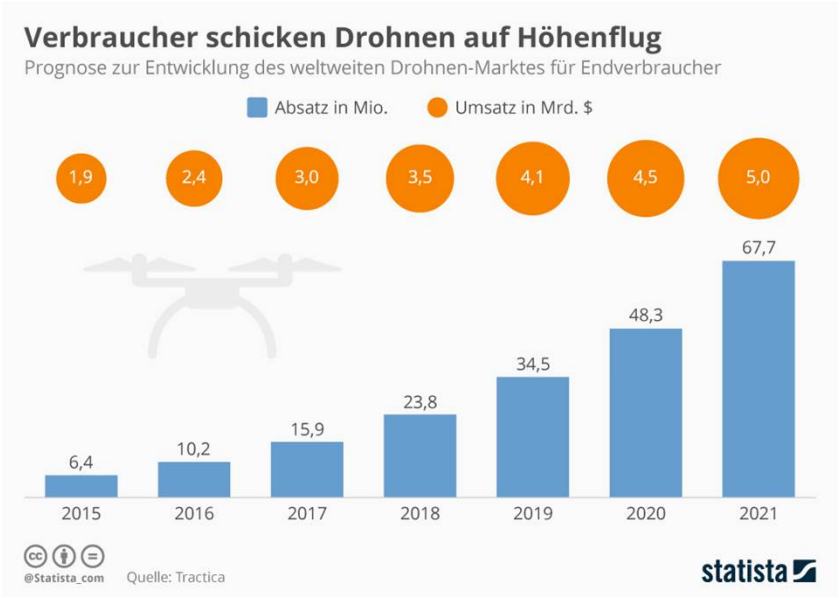


Abbildung 1.1: Prognose zur Entwicklung des weltweiten Drohnenmarktes[dro18]

1.1 Übersicht der Fähigkeiten aktueller Drohnen am Beispiel DJI

Um einen besseren Überblick über die Fähigkeiten aktueller Drohnen zu haben, folgt hier eine Vorstellung der neuesten Drohnenmodi am Beispiel von der neuesten DJI Drohne Phantom 4 Pro V2.0. Diese wird aktuell als fortschrittlichste Drohne im Amateurbereich angesehen. Die Kamera besteht aus einem speziell entworfenen F2.8 Weitwinkelobjektiv mit einer Brennweite von 24mm bestehend aus acht Elementen und einem 1-Zoll CMOS Sensor mit 20 Megapixelkamera und einer mechanischen Blende, die Verzerrungen eleminiert. Das Videoverarbeitungssystem unterstützt H.264 in 4k mit 60 fps Aufnahmen oder H.265 Videos mit einer Framerate von 30 fps. Unterstützt wird die Drohne von einem Netzwerk an Sensoren, welches FlightAutonomy genannt wird. Dies beinhaltet die Primäre Kamera, ein Infrarot-Erfassungssystem, Dual-Band Satellitenpositionierung durch GPS und GLONASS, redundante IMU und Kompasse, zwei Ultraschall-Entfernungsmesser und leistungsfähige Computerkerne. Die sechs Sichtsensoren sind in jeweils drei Paar angeordnet, die konstant die relative Geschwindigkeit zwischen der Drohne und Objekten berechnet. Das Infrarot-Erfassungssystem nutzt die Drohne um ein 3D-Scanning durchzuführen und somit Kollisionen zu vermeiden, indem sie die Abstände zu Hindernissen misst. Die Ultraschall-Sensoren unterstützten dies. Aufgrund dieses Sensornetzwerkes bietet DJI verschiedene Flugmodi an, die im Folgenden kurz vorgestellt werden.[dji18d]

1.1.1 TapFly-Modus

Im TapFly-Modus kann der User auf dem Bildschirm seines Smartphones auf einen Punkt in der Liveübertragung tippen und die Geschwindigkeit festlegen. Nachdem er den Start-Button betätigt hat, fliegt die Drohne autonom auf den jeweiligen Gegenstand zu. Hierfür verwendet die Kamera einerseits die Hauptkamera, andererseits die Infrarot- und Sichtsensoren zur Kollisionsvermeidung. Will der User während des Flugs ein anderes Objekt ansteuern, so kann er dies durch erneutes Tippen auf das neue Objekt verwirklichen. Somit ist dies eine autonome und maximal einfache Steuerung der Drohne, bei der der User sich hauptsächlich auf das Drehen der Drohne und die Steuerung der Kamera fixieren kann.[dji18d, dji18c]

1.1.2 FollowMe-Modus

Der DJI FollowMe Modus verfolgt eine Person oder ein Objekt auf Basis von GPS. Hier versucht die Drohne, das zu verfolgende Objekt im Mittelpunkt des Videos zu zentrieren. Hierfür muss die Drohne mit einer Fernbedienung oder idealerweise mit einem Smartphone verbunden sein. Die Drohne richtet sich dann aufgrund der GPS Position des Smartphones aus und folgt dem Objekt autonom. Hierfür kann sie in verschiedenen Flughöhen von 10 bis zu 500 m fliegen. Aufgrund der geringen Genauigkeit von GPS funktioniert dieser Modus nicht immer einwandfrei.[dji18d, dji18a]

1.1.3 Gesten-Modus

Die neue Phantom 4 V2.0 bietet dem User einen Gesten-Modus. Hier schwebt die Drohne einige Meter über dem User und fokussiert diesen mit der Hauptkamera. Anschließend kann der User mittels Heben der Arme der Drohne signalisieren, dass ein Foto gemacht werden soll. Daraufhin fokussiert die Kamera der Drohne das Ziel im Zentrum des Bildes. Anschließend streckt der User die Arme aus. Daraufhin startet die Drohne einen Countdown von drei Sekunden und macht daraufhin ein Bild der Zielperson.[dji18d]

1.1.4 Waypoints-Modus

Im Waypoints-Modus kann der User auf einer Karte verschiedene Wegpunkte festlegen, die die Drohne später autonom abfliegen soll. Hierfür müssen als erstes die Wegpunkte gesetzt werden und die Route einmal selbst abgeflogen werden. Anschließend kann die Drohnen die Route autonom selbst abfliegen, während der User die Drohne rotieren und die Kamera steuern kann. Auch können Routen abgespeichert und zu einen späteren Zeitpunkt erneut abgeflogen werden.
Eine Erweiterung dieses Algorithmus ist der Draw-Modus, in dem der User auf einem Smartphone auf dem übertragenen Livebild eine Flugroute einzeichnet. Hier kann der User zwischen den Forward-Mode, in dem die Kamera stets in Flugrichtung

ausgerichtet ist, und dem Free-Mode, in dem der User die Drohe frei um die eigene Achse rotieren kann, wählen. Dies ermöglicht dem User autonom Flugrouten abzufliegen und sich somit auf das Rotieren der Drohne und das Aufnehmen von Filmen zu konzentrieren.[dji18d, dji18b]

1.1.5 ActiveTrack-Modus

Der ActiveTrack-Modus ist eine Weiterentwicklung des FollowMe-Modus aus Kapitel 1.1.2. Hier verwendet die Drohne alle oben genannten Sensoren um ein Objekt zu verfolgen. Hierfür wird die Hauptkamera verwendet, um das Objekt zu verfolgen und zu filmen. Es muss zu Beginn in der DJI GO 4-App zunächst ein Kästchen um das zu verfolgende Objekt gezogen werden. Wird das ActiveTrack gestartet, richtet sich die Drohne stets so aus, dass das zu verfolgende Objekt sich im Mittelpunkt des Bildes befindet. Weiterhin verfolgt die Kamera die Person in einem bestimmten Abstand und kann diese sogar umkreisen. Hierfür verwendet die Kamera die Infrarot- und Sichtsensoren, sowie das GPS für die Abstandsmessung zur Person sowie zur Kollisionsvermeidung. Laut DJI ist die Drohne sogar in der Lage, das Objekt wieder zu finden, wenn es kurz aus dem Sichtfeld der Kamera geraten ist. Für ein möglichst gut funktionierendes ActiveTrack empfiehlt DJI, dass das Objekt möglichst kontrastreich zum Hintergrund ist. Hierbei bietet DJI drei Modi an. Im Profile-Modus verfolgt die Drohne das Objekt in dem es hinter oder neben ihm fliegt. Im Spotlight-Mode kann der User eine beliebige Flugroute steuern, die Drohne richtet jedoch automatisch ihre Hauptkamera auf das zu verfolgende Objekt aus. Im Circle-Mode fliegt die Drohne einen Kreis um das zu verfolgende Objekt und zentriert die Kamera auf das zu verfolgende Objekt.[dji18e]

All diese Features beweisen die zunehmende Autonomie der Drohnen und die bestehenden Möglichkeiten den User zu unterstützen, dass dieser selbst mit wenig Erfahrung anspruchsvolle Flugmanöver durchführen kann. Auch ist dies ein Beweis für die verbesserten Regelungsalgorithmen der Drohnen. Diese werden zukünftig noch besser werden. Jedoch bieten diese Entwicklungen auch Gefahren, da die Benutzer entweder durch fehlende Erfahrung oder durch Vorsatz Ziele aus der Luft überwachen, gefährden oder angreifen können.

1.2 Gefährdungen durch Drohnen

Um die ausgehende Gefahr von Drohnen einordnen zu können, benötigt es eine Eingliederung der Gefahren. Das Fraunhofer-**I**nstitut für **O**ptronik, **S**ystemtechnick und **B**ildauswertung (IOSB) teilt das Missbrauchspotenzial kleiner UAVs grob in drei Kategorien ein:

1. Störung
2. Gesetzeswidrigkeit

3. Gefährdung

In die erste Kategorie fallen Ausspähungen und unerlaubtes Erkunden von bestimmten Bereichen oder das gezielte Ablenken. Zur zweiten Kategorie gehören unter anderem Schmuggel oder Diebstähle von verschiedenen Objekten wie z.B. Waffen oder Drogen, Spionage, das Fliegen in Flugverbotszonen, wie z.B. an Flughäfen, militärischen Gebieten oder über Justizvollzugsanstalten. In die letzte Kategorie gehören schließlich das Durchführen von Anschlägen auf Personen, Fahrzeuge, Flugzeuge oder Gebäuden[IT16]. Ein Beispiel für die Gefahr, die von UAVs ausgeht, haben kürzlich Greenpeace-Aktivisten bewiesen, die eine als Superman verkleidete Drohne gegen ein Atomkraftwerk gesteuert haben[gre18]. Alleine dieses Beispiel lässt die Notwendigkeit der Entwicklung von Abwehrsystemen von UAVs erkennen.

1.3 Bestehende Systeme zur Drohnenabwehr

Um auf die bestehende Gefahr von Drohnen reagieren zu können, gibt es auf dem Markt bereits verschiedenen Abwehrsysteme. Diese Abwehrsysteme nutzen verschiedene Sensoren und Aktuatoren um Drohnen zu erkennen, zu verfolgen und abzuwehren. Dafür werden diese in die drei Teilbereiche Detektion, Verifikation und Intervention eingeteilt. Ein Vergleich verschiedener Detektionssysteme ist in Tabelle 1.1 zu sehen.

Das DLR ist mit verschiedenen Instituten an der Entwicklung eines mobilen Multisensorgesamtsystems zur Drohnenabwehr beteiligt, das Institut für Technische Physik in Stuttgart ist hierbei für die Bereiche Detektion und Verifikation zuständig. Eine Gesamtübersicht des Systems ist in Abbildung 1.2 zu sehen. In der Detektionseinheit dieses Systems, soll durch ein Multisensorsystem bestehend aus Radar und einer 360° Kamera eine grobe Positionsbestimmung erfolgen und diese an eine **S**chwenk- **u**nd **N**eigeplattform (SUN) weitergegeben werden, welche eine 3D Lokalisierung mit verschiedenen Sensoren, bestehend aus Laser, Kamera und Trackingsystem, durchführt. Durch diese genaue Ziel- und Positionsbestimmung soll einerseits das Sendersignal der Steuerung gejammt werden und andererseits soll durch einen GPS Spoofer die Kontrolle über das UAV erlangt werden, indem das UAV gezielt durch falsche GPS-Daten beeinträchtigt wird. Somit kann das Gefahrenobjekt gewünscht gesteuert und gelandet werden.

Teil des Tracking, Imaging und Laser Sensor-Trailers soll ein 360° Kamerasystem werden, das zur Bestimmung der Raumwinkel, also Azimut und Elevation, dienen soll und somit die genaue Ausrichtung eines Lasers für Laserranging ermöglichen soll. Der Vorteil dieses 360° Kamerasystems ist eine dauerhafte Rundumsicht, d.h. es kann stets der ganze Himmel mittels eines Panoramabildes überwacht werden und somit alle interessanten Objekte erkannt werden. Somit kann ein User aus den erkannten Objekten auswählen, welches dieser Objekte weiterhin verfolgt werden soll. Anschließend wird für dieses Objekt eine genauere Positionsbestimmung durchgeführt.[Sul17, Kü16, Lot16, IT16]

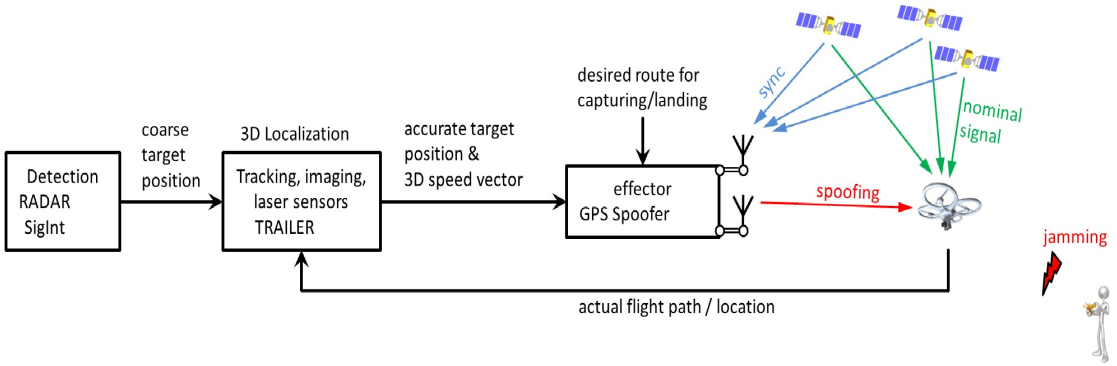


Abbildung 1.2: Gesamtsystemübersicht des Projektes

In folgender Arbeit werden zuerst die Projektziele und Anforderungen an das System in Kapitel 2 definiert, dann wird die Verarbeitungskette des Systems in Kapitel 3 vorgestellt. Anschließend werden die verwendeten Bildverarbeitungsalgorithmen in Kapitel 4 erklärt und der Programmaufbau in Kapitel 5 dargestellt. Abschließend erfolgen Validierungstests des Systems in Kapitel 6 und ein Fazit und Ausblick in Kapitel 7.

Detektionsmethode	Vorteile	Nachteile
Akustische Sensoren	• Möglichkeit der Erkennung autonomer UAV • Billig/ Passiv	• kurze Detektionsweite/ -höhe • Lärm kann System stören
Magnetometer	• Passiv • erkennt autonome UAV	• relativ kurze Detektionsweite/ -höhe
Laser	• erkennt selbst kleine Objekte	• Reichweite von wenigen 100 Metern
Radar	• Detektion über weite Entfernungen • kann große, schnelle Objekte erkennen	• Sehr hohe Kosten • erkennt kleine Objekte schlecht
Radio Frequency Sensor	• gute Reichweite • kann UAV und Operator erkennen durch Quelle/Empfänger- oder GPS-Signalverfolgung	• erkennt keine autonomen UAV
Electro-Optical/ Infrared Sensor	• Passiv • keine Limitierung durch Gebäude und Objekten	• hohe Kosten • relativ kurze Detektionsreichweite

Tabelle 1.1: Vergleich verschiedener Detektionsmethoden[Sul17]

2 Projektziele und Anforderungen

Im Folgendem werden die Projektziele und Projektanforderungen definiert, an denen sich der Erfolg des Projektes messen lässt.

2.1 Definition der Projektziele

Das Projekt wird in Haupt- und Nebenzielen unterteilt. Hauptziele sind von wichtiger Bedeutung, während Nebenziele weniger Priorität geniesen.

2.1.1 Hauptziele

1. Detektion von Drohnen und Flugzeugen mit Hilfe der Ladybug 5+ Kamera auf einer Entfernung von mindestens 30 Metern
2. Kontinuierliche Weitergabe des Azimuth und Elevationswinkels des Zielobjekts an ein Labview Programm
3. Realtime-Fähigkeit des Systems für die Verfolgung schneller Drohnen

2.1.2 Nebenziele

1. Möglichkeit der gleichzeitigen Detektion von mindestens 3 Drohnen
2. Repräsentative Nutzungsmöglichkeit der Software für Demonstrationszwecke bei Messen oder externen Finanzierern
3. Erstellung einer ausführlichen Dokumentation zur Weiternutzung der Software für spätere Arbeiten
4. Erstellung eines Posters für die Öffentlichkeitsarbeit

2.2 Anforderungen des Projekts

Im Folgenden werden die Anforderungen an das Projekt definiert, die zu einer erfolgreichen Umsetzung der Projektziele nötig sind.

2.2.1 Software Anforderungen

Die Anforderungen an die Software sind in Tabelle 2.1 aufgeführt.

Nummer	Anforderung	muss/ kann-Kriterium	erreicht/ nicht erreicht
1000	Einbindung des SDK von Ladybug in C++	muss	erreicht
1050	Einbindung von OpenCV in C++	muss	erreicht
1100	Programmieren und Testen von unterschiedlichen Bildverarbeitungsalgorithmen für die Detektion von Drohnen	muss	erreicht
1150	Weitergabe der Positionsdaten des Schwerpunktes der detektierten Drohnen an eine geeignete Schnittstelle für ein Labview Programm	muss	erreicht
1200	Verarbeitung der Bilder mit einer Rate von mindestens 10 Hz	muss	nicht erreicht
1250	Minimierung der Delayzeit zwischen Bildaufnahme und Weitergabe der Winkel von maximal 30 ms	muss	nicht erreicht
1300	Verfolgung von mindestens 3 Drohen mit passender Zuordnung zwischen den Bildern	kann	erreicht
1350	Optische Markierung der letzten 5 Flugpunkte einzelner Drohnen mittels Punkte im Bild	kann	erreicht
1400	Auswahl sinnvoller Voreinstellungen für die Detektionsalgorithmen	muss	erreicht
1450	Auswahl eines robusten Algorithmus und Parametersets für die Bereiche Halle, Hörsaal und Freifeld	muss	erreicht
1500	Auswahlmöglichkeit per Mausklick für die Weitergabe der Positionsdaten an Labview	kann	erreicht
1550	Threadbasierter Programmaufbau zur Minimierung der Delayzeiten	muss	erreicht
1600	Möglichkeit eine Region of Interest im Programm einzustellen	kann	erreicht
1650	Möglichkeit des Ladens eines externen Schablonenbildes, dass nicht interessante Regionen ausschließt	muss	erreicht

Tabelle 2.1: Softwareanforderungen

2.2.2 Anforderungen an die Grafische Oberfläche

Die Anforderungen an die grafische Oberfläche sind in Tabelle 2.2 dargestellt.

Nummer	Anforderung	muss/ kann-Kriterium	erreicht/ nicht erreicht
2000	Möglichkeit der Zuwahl und Abwahl verschiedener Bildverarbeitungsalgorithmen	muss	erreicht
2050	Auswahlmöglichkeit verschiedener Verfahren der einzelnen Bildverarbeitungsalgorithmen	muss	erreicht
2100	Möglichkeit der Veränderung der Parameter verschiedener Bildverarbeitungsalgorithmen mittels UI-Slider	muss	erreicht
2150	Anzeige des Panoramabildes der Kamera mit mindestens 2 Hz	muss	erreicht
2200	Anzeige des Threshold Images mit mindestens 2 Hz	kann	erreicht
2250	Markierung der detektierten Regionen im Panoramabild	muss	erreicht
2300	Möglichkeit der Anzeige von jeweils einem Thumbnail von mindestens den drei größten erkannten Objekten	kann	erreicht
2350	Anzeige der Winkel der verschiedenen Objekte in einem Textfeld	muss	erreicht
2400	Anzeige der fps-Raten der Kamera in einen Textfeld	kann	erreicht
2450	Speichermöglichkeit der verwendeten Einstellungen des Programms	kann	erreicht

Tabelle 2.2: Anforderungen an die grafische Oberfläche

2.2.3 Anforderungen an die Schnittstellen

Die Anforderungen an die Schnittstellen sind in Tabelle 2.3 definiert.

Nummer	Anforderung	muss/ kann-Kriterium	Erreicht/ nicht erreicht
3000	Auslesen der Ladybug5+ Kamera mittels Ladybug SDK und USB 3.0	muss	erreicht
3050	Zwischenspeichern der Panoramabilder auf gemeinsamen Ringbuffer	muss	erreicht
3100	Zwischenspeicher der bildverarbeiteten Bild auf gemeinsamen Ringbuffer	muss	erreicht
3150	Übergabe der Positionsdaten an die TCP/IP-Socket-Schnittstelle	muss	erreicht
3200	Bereitstellung der programmierten Funktionen als Bibliothek und Einbindung in Labview	kann	nicht erreicht

Tabelle 2.3: Anforderungen an die Schnittstellen

2.2.4 Sonstige Anforderungen

Die sonstigen Anforderungen sind in Tabelle 2.4 dargelegt.

Nummer	Anforderung	muss/ kann-Kriterium	Erreicht/ nicht erreicht
4000	Bewertung der Leistungsfähigkeit der Detektionsalgorithmen in Bezug auf Größe und Entfernung der Drohnen	muss	erreicht
4050	Erstellung einer ausführlichen Dokumentation	muss	erreicht
4100	Erstellung eines Datenblattes des Gesamtsystems	kann	erreicht
4150	Erstellung einer Abschlusspräsentation	kann	erreicht
4200	Erstellung eines Informationsposters des Projekts	kann	nicht erreicht
4250	Teilnahme an einer Konferenz mit Präsentation und eventuell Vorführung der Ergebnisse	kann	erreicht

Tabelle 2.4: Sonstige Anforderungen

3 Toolchain

Die Detektionseinheit setzt sich aus mehreren unterschiedlichen Komponenten zusammen. Wie in Abbildung 3.1 zu erkennen, ist die Ladybug 5+ Kamera über ein USB3 Kabel an einen Hochleistungsrechner angeschlossen, auf den das Programm ausgeführt wird. Dieses Programm verwendet die Bibliotheken LadybugSDK, OpenCV, CUDA und Qt für verschiedene Funktionen. Abschließend werden die Winkel, die vom Programm ausgewertet wurden, via TCP-IP an das Nachfolgesystem SUN geschickt. In folgenden Kapitel wird auf die einzelnen Komponenten des Systems eingegangen und deren Funktion erläutert.

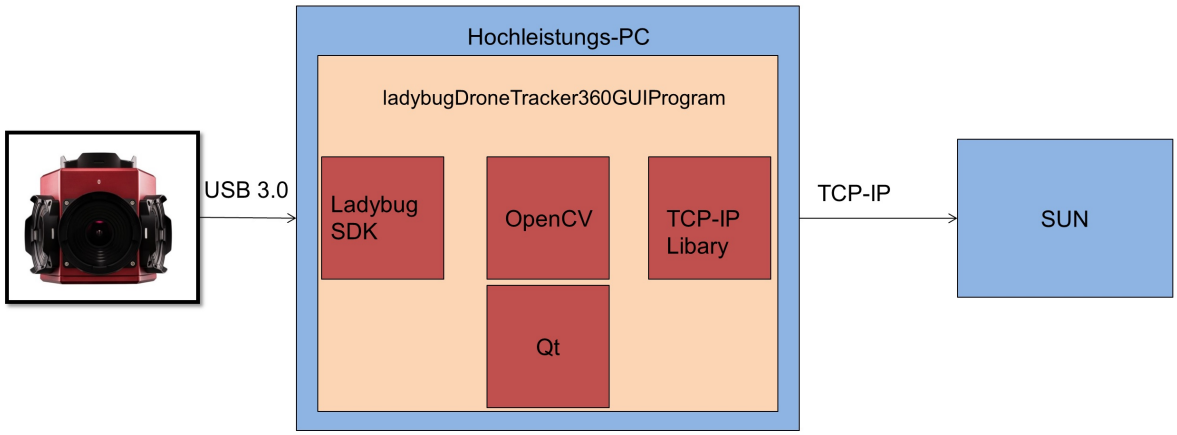


Abbildung 3.1: Systemübersicht der Toolchain

3.1 Ladybug 5+ Kamera

Als Detektionssensor für das System wurde die FLIR Ladybug 5+ ausgewählt, da diese Kamera mehrere Vorteile hat, die im Folgenden erklärt werden.
FLIR bietet mit der Kamera ein Ladybug **S**oftware **D**evelopment **K**it (SDK) an, welches viele Funktionen und Beispiele für das Kamerasystem anbietet. Somit kann die Kamerasteuerung mittels dieses SDK erfolgen und muss nicht aufwendig selbst umgesetzt werden. Außerdem profitiert man von den bereits vorhandenen Programmbeispiel, weil das Aufnehmen eines Videos und viele andere Funktionen bereits als vorkompilierte Programme vorhanden sind, der Sourcecode jedoch gleichzeit zugänglich ist. Auch bietet FLIR einen sehr guten Support für das Kamerasystem, wodurch Rückfragen und Problemen schnell gelöst werden können, was aufgrund der begrenzten Zeit

des Projektes von Vorteil ist. Das Kamerasystem ist mit USB3 an den Computer angeschlossen, wodurch eine hohe Übertragungsrate zum Computer sichergestellt ist. Da die Panoramabilder eine Größe von 30 MP haben, kann so durch eine reibungslose Übertragungsrate von bis zu 120 MB/s sichergestellt werden, dass die Bilder schnell auf dem Computer zur weiteren Verarbeitung angelangen. Auch wirbt die Kamera mit einem pixeltreuen Koordinatensystem, wodurch eine einfache Positionsbestimmung möglich ist. Abschließend bietet der Markt für 360° Kamerasysteme nur wenig Auswahl und diese Kamera liefert die nötige Verarbeitung und Ausgereiftheit, um Drohnendetektion durchzuführen. Sonstige Systeme sind auf dem Markt quasi nicht existent oder haben nicht die nötige Professionalität. In Tabelle 3.1 sind die physikalischen Eigenschaften der Kamera angegeben.[lad18c, lad18a, lad18b]

FLIR Ladybug 5+	
Auflösung	2048 x 2448
Bildrate	10 FPS JPEG Komprimiert; 5 FPS unkomprimiert
Megapixels	30 MP (5 MP x sechs Sensoren)
Sensor	Sony ICX655 CCD x 6, CCD, 2/3
Auslesemethode	Globaler Shutter
Pixelgröße	3.45 µm
Optik	6 high quality Linsen mit 4.4 mm Brennweite
Verstärkungsbereich	0 dB bis 18 dB
Field of View	90% einer Kugel
Analog-/ **D**igital **C**onverter (ADC)	12-bit
Bildverarbeitung	Shutter, gain, Weißabgleich, gamma und JPEG Komprimierung (programmierbar via Software)
Flash Speicher	1 MB non-volatile memory
Schnittstelle	USB3
Betriebsspannung	12 V - 24 V
Energieverbrauch	24 W
Abmessungen	197 mm Durchmesser, 160 mm Höhe
Masse	3.0 kg

Tabelle 3.1: Eigenschaften der Ladybug 5+[lad18b]

3.2 Hochleistungscomputer und Grafikkarte

Bildverarbeitungsalgorithmen, die möglichst schnell ausgeführt werden sollen, sind entweder für die Grafikkarte oder auch für den Prozessor sehr anspruchsvoll. Da die Kamera, laut Kapitel 3.1, sehr große Bilder mit einer 30 MP Auflösung liefert, müssen die Bildverarbeitungsalgorithmen auf einer sehr großen Anzahl an Pixel angewandt werden. Weiterhin erfolgt das Stichting der Panoramabilder der Ladybug 5+ Kamera auf der Grafikkarte. Somit wurde sich nach einigen Performancetests auf den Bürorechner des Instituts dafür entschieden, den momentan leistungsstärksten PC-Tower des DLR-Shops zu erwerben, um eine möglichst schnelle Performance des Programms zu erreichen. Hierfür wurde ein 4-Kern-Prozessor vom Typ Intel W-2125 ausgewählt, mit einer Taktrate von bis zu 4,0 GHz. Dieser Prozessor besticht durch eine hohe Rechenleistung und die Unterstützung von Multithreading durch seine vier Kerne. Um ein möglichst schnelles Stitching zu garantieren und die Möglichkeit zu haben, Bildverarbeitungsalgorithmen mit CUDA auf der Grafikkarte auszuführen, wurde zusätzlich eine NVIDIA Quadro P4000 Grafikkarte in den PC eingebaut. Die Leistungsangaben des Rechners sowie der Grafikkarte sind in Tabelle 3.2 zu sehen:

Dell Precision Tower 5820 Workstation	
Prozessor	Intel Xeon W-2125 4,0 GHz Turboboost, 4 Kerne, 8,25 MB Cache, Hyper-Threading, DDR4-2666
Arbeitsspeicher	32 GB DDR4-RDIMM-Speicher (4x8 GB), 2.666MHz, ECC
Betriebssystem	Windows 10 Pro
NVIDIA Quadro P4000	
CUDA Parallel Processing Cores	1792
Peak Single Precision FP32 Performance	5.3 TFLOPS
GPU Speicher	8 GB GDDR5
Memory Interface	256-bit
Memory Bandwidth	243 GB/s
System Interface	PCI Express 3.0 x 16

Tabelle 3.2: Leistungsdaten des Hochleistungscomputers und der Grafikkarte

3.3 Ladybug SDK

Für jedes FLIR Ladybug-System gibt es ein umfangreiches SDK mit zahlreichen Funktionen zur Steuerung der Bilderfassung, Erzeugung von sphärischen oder Panorama-Bildern und Ändern der Kameraeinstellungen. So liefert das Ladybug SDK das

Programm LadybugCapPro, weiteren Beispielquellcode für einen erleichterten Einstieg in die C/C++ Programmierumgebung, Kameratreiber und eine **A**pplication **P**rogramming **I**nterface (API)-Softwarebibliothek, die es dem Programmierer erleichtert die Ladybug-Funktionalitäten in eigenen Anwendungen zu integrieren. So bieten die API Codebeispiele für die Steuerung der Bildaufnahme und Datenformate, Steuerung der Kameraeinstellungen, wie z.B. Belichtungsautomatik, Auflösung und Framerate oder Konfiguration von GPIO-Trigger oder Stroposkop-Einstellungen. Auch verarbeitet die API die Bilder, kann diese hardwarebeschleunigt entzerren, aneinanderfügen und überblenden und rendert die Ausgabe von Panorama, Dome- oder Cube-Maps. Weiterhin können Bildbearbeitungsprozesse angewandt werden wie z.B. Farbinterpolierung, Belichtungskorrektur, Weißabgleich und Gamma- und Tone-Mapping. Somit lohnt sich die Verwendung des SDK in dieser Arbeit, da somit die Ansteuerung der Kamera deutlicht vereinfacht wird und von den oben genannten Funktionen profitiert werden kann.[lad18c]

3.4 OpenCV

Die weitverbreiteste Computer Vision Bibliothek ist OpenCV. Diese wird verwendet um die Bildverarbeitung in diesen Projekt durchzuführen. Diese Bibliothek ist unter einer BSD-Lizenz veröfftenlicht und dadurch frei erhältich für den akademischen und gewerblichen Gebrauch. Auch sind ein C++, Phython und Java Interface verwirklicht und sie unterstüzt die weitverbreitesten Betriebssysteme wie Windows, Linux, macOS, iOS und Android. Mittlerweile verfügt OpenCV über mehr als 2500 optimierte Algorithmen. Diese decken sowohl grundlegende Bildverarbeitungsfunktionen, wie z.B. Umwandlungen von Bildern in verschiedene Farbräume ab, aber auch komplexere Algorithmen zur Gesichtserkennung oder zur Erstellung von 3D-Punktewolken mittels Stereokameras. Diese Bibliothek ist auf Recheneffizienz ausgelegt um den Gebrauch für Echtzeitanwendungen zu unterstützen. Da die Bibliothek in C/C++ geschrieben wurde, unterstützt sie Multicoreprozessierung. Weiterhin bietet die Bibliothek eine CUDA Bibliothek, mittels dieser Bildverarbeitungsoperationen auf einer NVIDIA Grafikkarte vollzogen werden können, was oft einen Geschwindigkeitsgewinn im Vergleich zur gleichen Operation auf der CPU bewirkt, da Bildverarbeitungsoperationen oft sehr viele gleiche parallele Operationen sind, die auf einer Grafikkarte schneller verarbeitet werden können.[ope18b, ope18a, cud18b]

3.5 Qt

Qt ist ein plattformübergreifendes GUI-Framework für die Entwicklung von Benutzeranwendungen. Windows-, Mac- und Linux-Programme lassen sich damit ebenso entwickeln wie Apps für Android, iOS und Windows Phone. Ein großer Vorteil von Qt ist, dass durch den Signal-Slot-Ansatz auf Benutzerinteraktion reagiert werden kann. Hierbei löst das jeweilige Objekt auf der Benutzeroberfläche, wenn der Benut-

zer dieses anklickt oder ändert, ein Signal aus. Jedem Signal wird ein zugehöriger Slot zugewiesen, in dem die Reaktion des Programms auf die Benutzerinteraktion definiert ist. Somit kann mittels Multithreading-Programmen der zugehörige Programmablauf dementsprechend reagieren, ohne das Gesamtprogramm unterbrechen zu müssen. Das Multithreading ist gleichzeitig ein weiterer Vorteil. Dadurch, dass Klassen bestimmten Threads zugewiesen werden können, können diese Threads parallel ausgeführt werden, was auf Multicore-Prozessoren zu einer besseren Performance führen kann als bei einem sequentiellen Programmablauf. Hierfür stellt Qt die Klasse QThread (vergl. [qtr18]) bereit. Diese Klasse bietet auch Threadhandling-Tools wie z.B. Semaphore (vergl. [qse18]) an, die dafür zuständig sind, Speicherinkonsitzenzen durch das Zugreifen mehrerer Threads auf den selben Speicher zu verhindern. Da das Drohnendetektionsprogramm einerseits ein User Interface anbieten soll, das auch auf Benutzereingaben reagiert und andererseits das Programm Multithreading unterstützen soll, um eine bessere Performance zu erreichen, wurde sich für Qt als Tool zur Umsetzung der GUI und des Multithreadings entschieden. Auf den Aufbau des Programms inklusive Multithreading und Semaphore wird in Kapitel 5 näher eingegangen.[qt18, qtp18, qse18, qtr18]

3.6 Cuda Toolkit

Das NVIDIA CUDA Toolkit stellt eine Entwicklungsumgebung für high performance GPU Applikationen bereit. Das Toolkit wird dazu benutzt um Applikationen zu optimieren und zu beschleunigen, in dem Prozesse auf die Grafikkarte ausgelagert werden. Hierfür ist es lediglich nötig, eine NVIDIA Grafikkarte in der Desktop-Workstation verbaut zu haben und sich das NVIDIA Toolkit herunterzuladen. Das Toolkit beinhaltet optimierte GPU Bibliotheken, Debugging und Optimierungstools, einen C/C++ Compiler und eine Runtime-Bibliothek. Da Grafikkarten in vielen Rechenoperationen schneller sind als CPUs, können viele Operationen ausgelagert werden, wie z.B. Operationen im Bereich Linearer Algebra, Bild und Videoverarbeitung, Deep Learning und Graphenanalyse. Auch stellt NVIDIA mit CUDA Integrationen für viele Programmiersprachen und einige Pakete, wie z.B. OpenCV, bereit. Ein weiterer Vorteil ist, dass CUDA Applikationen auf allen gängigen NVIDIA GPU-Familien benutzt werden können. Aufgrund der Größe der Bilder mit 30 MP, hat man sich dazu entschieden, die Bildverarbeitungsalgorithmen auf die Grafikkarte auszulagern, um so die Anwendung zu beschleunigen.[cud18b, cud18a, ope18a]

3.7 Interface zu SUN

Um die Schnittstelle zum nachfolgenden Programm, das die SUN steuert, sicherzustellen, wurde sich für eine **T**ransmission **C**ontrol **P**rotocol **/I**nternet **P**rotocol (TCP/IP)-Schnittstelle entschieden. TCP/IP ist ein Protokoll mit zwei Schichten. Die obere Schicht, Transmission Control Protocol, zerlegt eine Nachricht oder Datei in klei-

nere Pakete, welche über das Internet übertragen werden und von der DCP-Schicht des Empfängers wieder zusammengefügt werden. Die untere Schicht, Internet Protocol genannt, ist für die Adressierung jedes einzelnen Pakets zuständig, sodass diese zum richtigen Empfänger gelangen. TCP/IP verwendet für die Kommunikation das Client-Server-Model, wodurch von einem Client ein Dienst, also in diesen Fall das Empfangen zweier Winkel und der Zeitinformationen, angefordert wird und von einem anderen Computer, den sogenannten Server, dieser Dienst bereitgestellt wird. Dabei verläuft die Kommunikation primär von Punkt zu Punkt. Auch ist dieses Modell zustandslos, was bedeutet, dass jede Anfrage eines Clients als neue Anfrage ohne Bezug zu vorherigen behandelt wird. Der große Vorteil hiervon ist, dass das Netzwerk von dauerhaften Datenverkehr freigehalten wird. Dieses Protokoll kann einerseits zwischen zwei Computern, die im selben Netzwerk sind und auf denen TCP/IP eingerichtet ist, verwendet werden, oder es kann auch Daten von zwei Programmen, die auf den selben Computer ausgeführt werden, austauschen. Dies erfolgt, indem man als IP des Empfängers, die eigene Computer-IP angibt. Diese Schnittstelle wurde gewählt, da sowohl einerseits beide Programme auf den selben Computer ausgeführt werden sollen, andererseits auch ein Fernzugriff via Netzwerk auf die Winkelausgabe von Azimut und Elevation möglich sein soll. [tcp18a, tcp18b, Dre04]

4 Bildverarbeitung

Im folgenden Kapitel werden als Erstes verschiedene grundlegende Methoden zur Erkennung von Drohnen in Bildern vorgestellt, anschließend werden diese untereinander abgewägt und sich für eine Methode entschieden. Danach werden Bildverarbeitungsoperationen, die für das Programm verwendet wurden, vorgestellt. Im Anschluss werden die Algorithmen auf ihre Tauglichkeit zur Drohnendetektion evaluiert und ein passendes Algorithmen- und Parameterset ausgewählt.

4.1 Methoden zur Bewegungserkennung in Bildsequenzen

Die Kamera, die in Kapitel 3 ausgewählt wurde, ist ein statisches System, d.h. die Kamera wird auf eine möglichst stabile Halterung befestigt und während des Programmablaufs nicht in ihrer Position verändert. Aus dieser Grundbegebenheit ergeben sich mehrere Methoden der Bildsegmentierung. Unter Bildsegmentierung versteht man, die Bildvorverabeitung, mit dem Ziel, dass das Bild danach in interessante Bereiche, sogenannte **R**egions **O**f **I**nterest (ROIs), und uninterssante Bereiche aufgeteilt ist. Im Folgenden werden Bildsegmentierungsmethoden zur Drohnenerkennung, die von einem statistischen System verwendet werden können, vorgestellt.

4.1.1 Objektsegmentierung anhand der Differenzbildmethode

Eine Methode zur Bildsegmentierung ist die Differenzbildmethode. In dieser Methode, werden Veränderungen des Bildes zum vorherigen Bild betrachtet. Der Ablauf ist in Abbildung 4.1 dargestellt. Zuerst werden zwei aufeinanderfolgende Bilder in Grauwertbilder umgewandelt. Anschließend werden diese Grauwertbilder von einander abgezogen. Das Ergebnis ist ein Differenzbild. Auf dieses Bild wird ein Schwellenwertfilter gelegt, der alle Veränderungen größer als den Schwellenwert als weißes Pixel und alle anderen Pixel in schwarze Pixel umwandelt. Als Ergebnis man erhält am Ende der Differenzbildsegmentierung ein binarisiertes Schwarz-Weiß-Bild in dem ROIs weiß dargestellt sind.[Erh08, SR14, Ste08, dif18]

Die Vorteile dieser Methode sind:

- Einfache Methode zur Objektsegmentierung
- Relativ geringer Rechenaufwand, da einfache Operationen
- Erkennt sämtliche Bewegungsveränderungen im Bild

Die Nachteile dieser Methode sind:

- Keine Erkennung von statischen, interessanten Gegenständen
- Keine Tiefeninformationen
- Bildrauschen wird auch als Bewegung erkannt

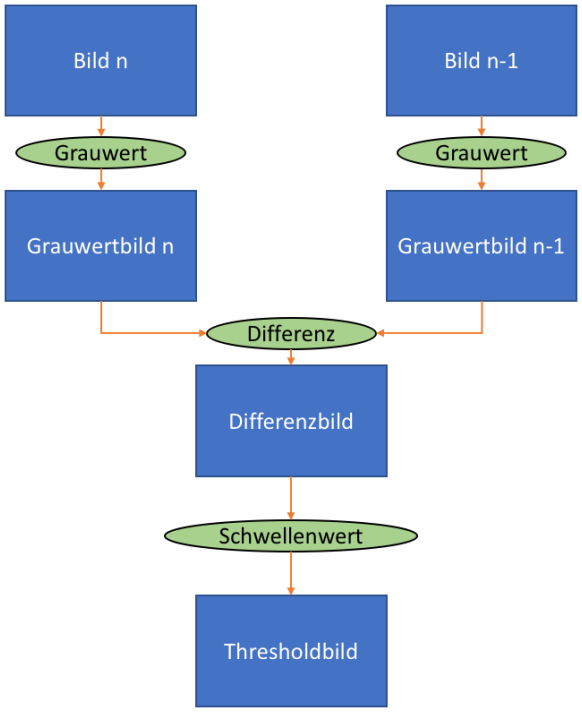


Abbildung 4.1: Ablaufdiagramm der Differenzbildmethode

4.1.2 Objektsegmentierung anhand der Transformation des Bildes in den HSV-Farbraum

Die Objektsegmentierung anhand der Transformation des Bildes in den **H**ue **S**aturation **V**alue (HSV)-Farbraum erfolgt durch Umwandlung der Rot, Grün und Blau Werte in den HSV-Raum und ein Vergleich der HSV-Werte mit einer vorher definierten Bereich des HSV. Das HSV-Modell wird bevorzugt, da es der menschlichen Wahrnehmung von Farbe ähnelt. Der HSV-Farbraum wird meist in Zylinderkoordinaten dargestellt, wobei der Farbwert(engl. Hue) den Winkel φ im Intervall von 0° bis 360° entspricht, die Farbsättigung(engl. saturation) im Intervall 0 bis 1 den Radius R entspricht und der Helligkeitswert (engl. value) im Intervall 0 bis 100% der Z-Koordinate entspricht. Die Darstellung des HSV-Farbraumes ist in Abbildung 4.2 zu sehen. Die Bildsegmentierung erfolgt dann aufgrund einer vorherigen Definition des interessanten HSV-Bereichs, d.h. es werden H_{min}, H_{max}, S_{min}, S_{max}, V_{min} und V_{max} definiert. Anschließend

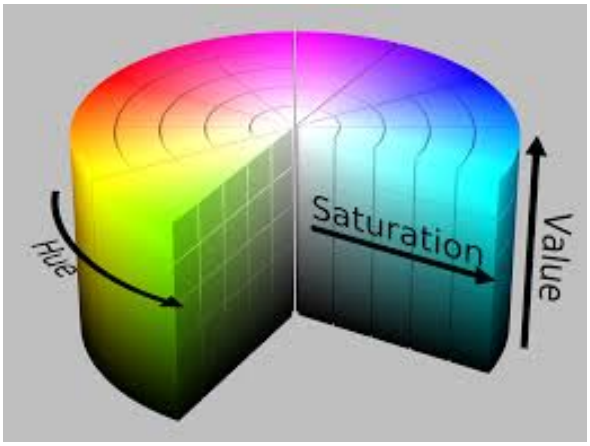


Abbildung 4.2: Darstellung des HSV-Farbraumes[Kri13]

wird jedes Pixel mit diesen Werten verglichen. Liegen die HSV-Werte innerhalb dieses Bereiches, so wird das Pixel als interessant betrachtet.[hsv18, Kri13, Kop07, Fol08] Die Vorteile dieser Methode sind:

- Relativ unanfällig für Helligkeitsänderungen
- Erkennung von statischen Objekten

Die Nachteile dieser Methode sind:

- Objektsegmentierung ist farbabhängig
- Uninteressante Objekte, die im HSV-Intervall liegen, werden ebenfalls erkannt
- Rechenaufwendig

4.1.3 Objektsegmentierung mit Hilfe von neuronalen Netzen

Die Objektsegmentierung mit Hilfe von neuronalen Netzen ist eine weitere Möglichkeit, interessante Bereiche aus Bildern zu filtern. Anfangs muss ein Modell gebildet werden, mit dem das neuronale Netzwerk zwischen Treffer und Niete entscheidet. Dies erfolgt, indem man dem Netzwerk eine große Menge an Bildern gleicher Größe zur Verfügung stellt, die als Treffer anzusehen sind. Das neuronale Netzwerk ist in mehrere Schichten unterteilt, wobei die Eingabeschicht das Bild entsprechend gewichtet und dann die Folgeschicht weiter gibt. Dies erfolgt über mehrere Schichten bis zur Ausgabeschicht, wobei jede zusätzliche Schicht den Rechenaufwand erhöht. In der Ausgabeschicht erfolgt dann eine Aussage, ob das Bild als Treffer oder Niete anzusehen ist. Das neuronale Netzwerk aktualisiert durch jedes Bild sein Modell. In der Anwendungsphase erkennt das neuronale Netzwerk Modell idealerweise mit einer sehr hohen Trefferwahrscheinlichkeit auch beim Vorlegen von neuen Bildern, ob das neue

Bild ein Treffer ist oder nicht. Somit kann eine Objekterkennung bzw. -segementierung erfolgen, indem das bestehende vorgelegte Bild der Kamera in Teilbereiche unterteilt und dann durch das neuronale Netzwerk untersucht werden kann.[neu18, Str97]

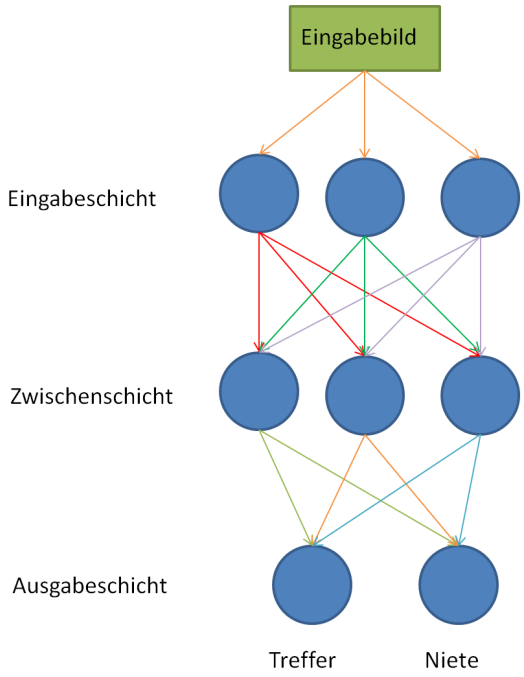


Abbildung 4.3: Aufbau eines neuronalen Netzes

Die Vorteile dieser Methode sind:

- Erkennung von stehenden und sich bewegenden Objekten
- Hohe Zuverlässigkeit durch Deep Learning

Die Nachteile dieser Methode sind:

- Bildabschnitte müssen alle die selbe Größe haben
- Hoher Rechenaufwand beim Lernen
- Ohne vorherige Bestimmung von interessanten Regionen schwer umsetzbar, da rechenaufwändig

4.1.4 Auswahl einer geeigneten Methode

Nach Abwägung der verschiedenen Vor- und Nachteile der beschriebenen Objektsegmentierungsmethoden in Kapitel 4.1.1, 4.1.2 und 4.1.3, hat man sich schließlich für die Umsetzung durch die Differenzbildmethode entschieden. Da Drohnen unterschiedlichster Farben erkannt werden sollen, ist die HSV-Methode ungeeignet. Die Bilder,

(a) Farbbild des 1. Bildes (b) Farbbild des 2. Bildes

Abbildung 4.4: Farbbilder des Beispiels

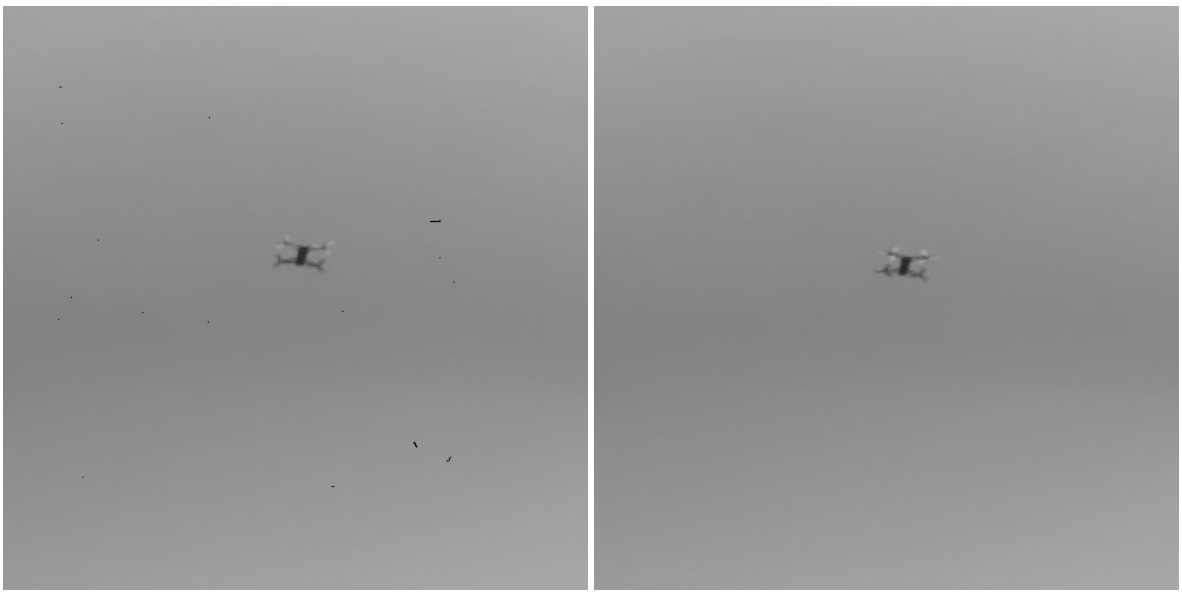
die die Kamera mit bis zu 30 MP ausgibt, sind sehr groß, wodurch ein neuronales Netzwerk nicht den gewünschten Erfolg bringen würde, da so das Bild in einzelne Bildsegmente unterteilt werden müsste und somit die Performance des Programms eher langsam wäre. Im folgenden Kapitel wird der genau Verlauf der Differenzbildmethode beschrieben.

4.2 Bewegungserkennung anhand der Differenzbildmethode

Bei der Differenzbildmethode werden zwei aufeinanderfolgende Bilder miteinander verglichen. Da die Verarbeitung der Bilder mit 10 Hz erfolgen soll (vergl. 2.2.1), wurde sich für die Differenzbildmethode mit zwei aufeinanderfolgenden Bildern entschieden. Es gibt auch Methoden, mehrere aufeinanderfolgende Bilder zu vergleichen (vergl. [MMP12]), diese wurden jedoch aufgrund der Größe der Bilder und der Performanceanforderungen nicht weiter verfolgt. Die einzelnen Schritte werden im Folgenden beschrieben. Die Beispielbilder als Farbbilder sind in Abbildung 4.4 zu sehen.

4.2.1 Erstellen eines Grauwertbildes

Um die beiden Bilder A und B mit den Pixelwerten a(r,g,b) und b(r,g,b) miteinander vergleichen zu können, werden diese Bilder als erstes in Bilder mit Grauwerten



(a) Grauwertbild des 1. Bildes (b) Grauwertbild des 2. Bildes

Abbildung 4.5: Grauwertbild Beispiele

transformiert. Dies erfolgt in OpenCV mit folgender Formel, wobei a_g und b_g die Grauwerte an der Stelle (x, y) der Grauwertmatritzen des jeweiligen Bildes sind und r,g und b die Rot-, Grün- und Blauwerte an der jeweiligen Stelle (x, y) der Farbmatritze des jeweiligen Bildes:

$$a_g(x, y) = 0.299r(x, y) + 0.587g(x, y) + 0.114b(x, y) \tag{4.1}$$

$$b_g(x, y) = 0.299r(x, y) + 0.587g(x, y) + 0.114b(x, y) \tag{4.2}$$

Ein Beispiel für die Grauwertbildung zweier Bilder ist in Abbildung 4.5 erkennbar.

4.2.2 Erstellen eines Differenzbildes

Um die Veränderungen von Bild A zu Bild B zu sehen, werden die beiden Grauwertbilder von einander abgezogen. Aus der Differenz der Grauwerte lässt sich herausfinden, ob sich ein Pixel im Vergleich zum vorherigen Bild geändert hat. Hier entsprechen alle Werte, die nicht 0 sind, grundsätzlich eine Veränderung. Die weitere Verarbeitung dieser Werte ist im nächsten Kapitel 4.2.3 beschrieben.

$$f_g(x, y) = a_g(x, y) - b_g(x, y) \tag{4.3}$$

4.2.3 Thresholding

Bei der Schwellenwertbildung wird auf den zuvor berechneten Differenzbild f(x,y) unter Vorgabe oder Berechnung einer Schwelle T ein Binärbild b(x,y) mit folgender Formel berechnet:

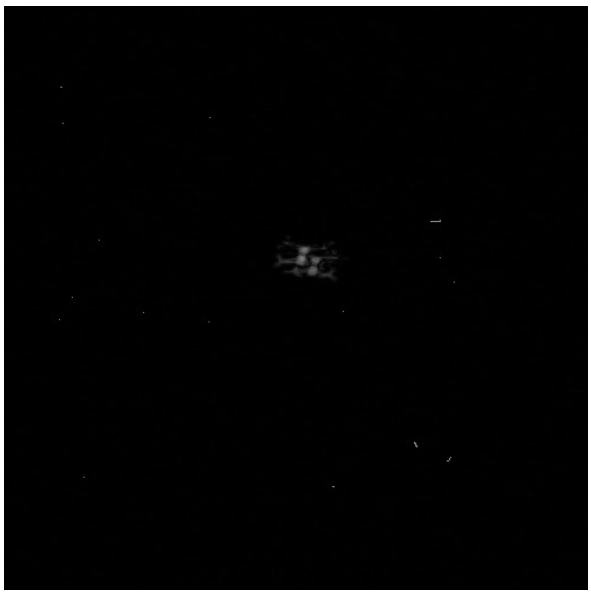


Abbildung 4.6: Differenzbild der beiden Bilder

$$b(x,y) = \begin{cases} 1, falls f_g(x,y) \geq T \\ 0, falls f_g(x,y) < T \end{cases} . \tag{4.4}$$

Am Ende dieser Operation erhält man eine biniarisierte Matrix, d.h. jedes Pixel in dieser Matrix ist entweder schwarz oder weiß. Dies benötigt man zur weiteren Bildverarbeitung. Ein Beispiel für das Bilden eines Binärbildes aus zwei aufeinanderfolgenden Bildern ist in Abbildung 4.7 dargestellt. Nach diesen Schritten ist das Bild segmentiert und alle interessanten ROIs liegen als weiße Regionen im Binärbild vor. Da jedoch z.B. durch Bildrauschen viele falsche Detektionen vorliegen, muss das Bild noch durch morphologische Operationen nachbearbeitet werden.

4.3 Morphologische Operationen

Morphologische Operationen sind grundlegende Bildverarbeitungsalgorithmen, die zur Beschreibung, Analyse und zur Veränderung von Formen in Bildern eingesetzt werden. Ihre Anwendung haben diese Operationen auf Binär- und Grauwertbildern. Grundaufgabe dieser Operationen ist es, Bildrauschen zu filtern, naheliegende Regionen zu verbinden und Falschdetektionen zu verhindern. Im Folgenden werden die grundlegenden morphologischen Operationen erklärt, die für die Verarbeitung des Differenzbildes in Betracht gezogen wurden, so wie deren unterschiedliche Kombinationen mathematisch erklärt und durch Beispiele angewandt. Abschließend erfolgt eine Evaluation der Algorithmen und es wird eine Kombination der Algorithmen bestimmt, die am besten für die Drohnendetektion geeignet ist. [Kar06, Ste08, SR14, Erh08]

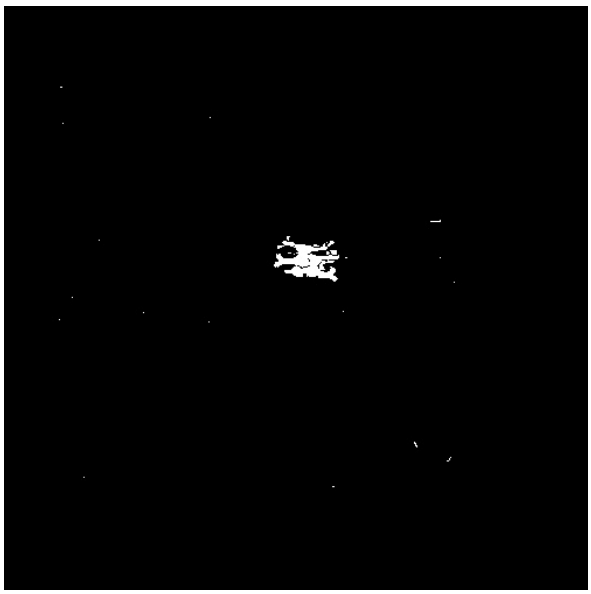


Abbildung 4.7: Thresholdbild der beiden Bilder

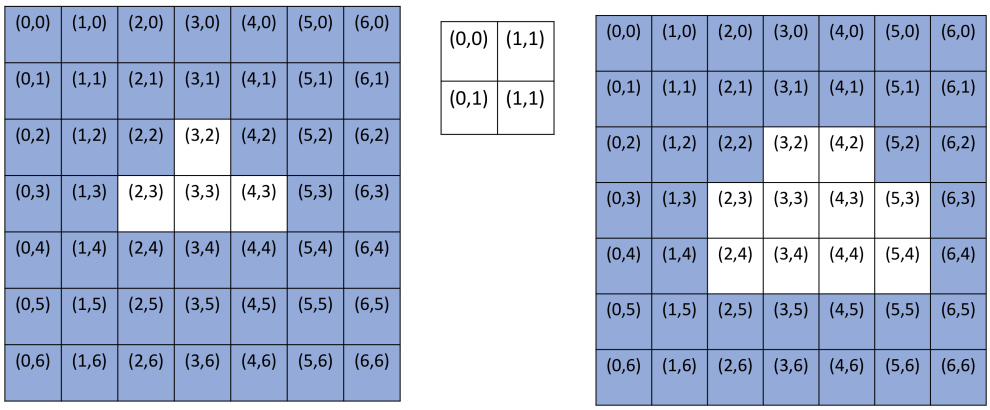
4.3.1 Dilation

Bei der Dilation werden zwei Punktmengen A und B mittels Minkowski-Addition miteinander addiert, wie in Formel 4.5 dargestellt. Hierbei erfolgt die paarweise Addition der Pixelkoordinaten der weißen Pixel des Bildes A mit dem Kernel B. Ein Beispiel ist in Abbildung 4.8 dargestellt.Hier sind die weißen Pixel in weiß dargestellt und die schwarzen Pixel in hellblau. Die Mengen A und B sowie die Lösungsmenge sind in Formel 4.6 angegeben.

$$A \oplus B = \{c \mid c = a + b, a \in A, b \in B\} \tag{4.5}$$

$$\begin{aligned} &A = \{(3,2);\ (2,3);\ (3,3);\ (4,3)\};\quad B = \{(0,0);\ (1,0);\ (0,1);\ (1,1)\} \\ &A \oplus B = \{(3,2);\ (2,3);\ (3,3);\ (4,3);\ (4,2);\ (5,4);\ (2,4);\ (3,4);\ (4,4);\ (5,3)\} \end{aligned} \tag{4.6}$$

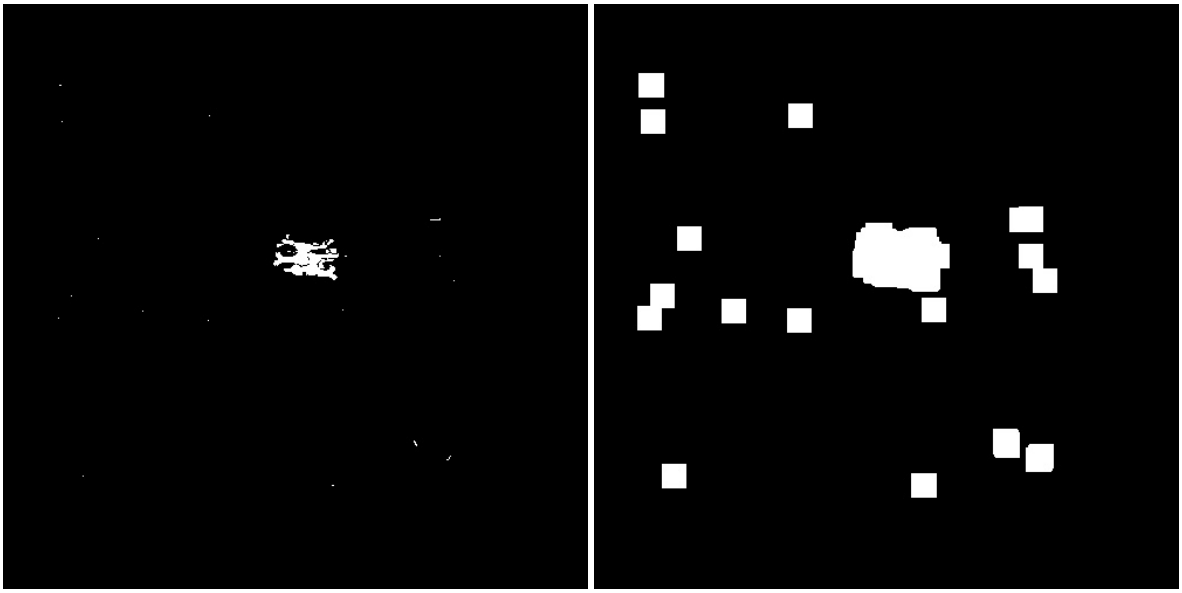
B ist das Strukturelement, mit dem dilatiert wird und ist in der Regel wesentlich kleiner als das Bild A. Eine Dilation des Bildes A mit dem Strukturelement B hat zur Folge, dass die Strukturen in Bild A verstärkt bzw. verdickt werden. Auch kann dies zu einer Vereinigung zweier Formen führen. Weiterhin können so Löcher und Risse geschlossen werden. Als Strukturelement werden oft um den Ursprung des Strukturelements angeordnete symetrische Formen verwendet. OpenCV bietet hier die Möglichkeit zwischen einer Elipse, einem Rechteck und einem Kreuz zu wählen. Ein Beispiel einer Dilation mit einem Quadrat der Seitenlänge 21 und dem Mittelpunkt (10/10) ist in Abbildung 4.9 dargestellt.[ope, Kar06, Ste08, SR14, Erh08]



(a) Beispielbild und Kernel

(b) Ergebnis der Minkowski-Addition

Abbildung 4.8: Beispiel einer Minkowski-Addition



(a) Threshold Bild

(b) Bild nach Anwendung der Dilation

Abbildung 4.9: Beispiel für eine Dilation

4.3.2 Erosion

Die Erosion ist das Gegenteil der Dilation aus Kapitel 4.3.1. Bei der Erosion werden zwei Punktmengen A und B mittels Minkowski-Subtraktion wie in Formel 4.7 miteinander subtrahiert. Hierbei werden die Pixelkoordinaten der weißen Pixel des Bildes A paarweise mit den Pixelkoordinaten des Kernels subtrahiert. Ein Beispiel ist in Abbildung 4.10 zu sehen. In diesem Fall sind die weißen Pixel in weiß dargestellt und die schwarzen Pixel in hellblau. Die Mengen A und B sowie die Lösungsmenge sind

in Formel 4.8 angegeben.

$$A \ominus B = \{c \mid c = c + b \in A, b \in B\} \tag{4.7}$$

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)
(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)

(0,0)	(1,1)
(0,1)	(1,1)

(a) Beispielbild und Kernel

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)
(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)

(b) Ergebnis der Minkowski-Subtraktion

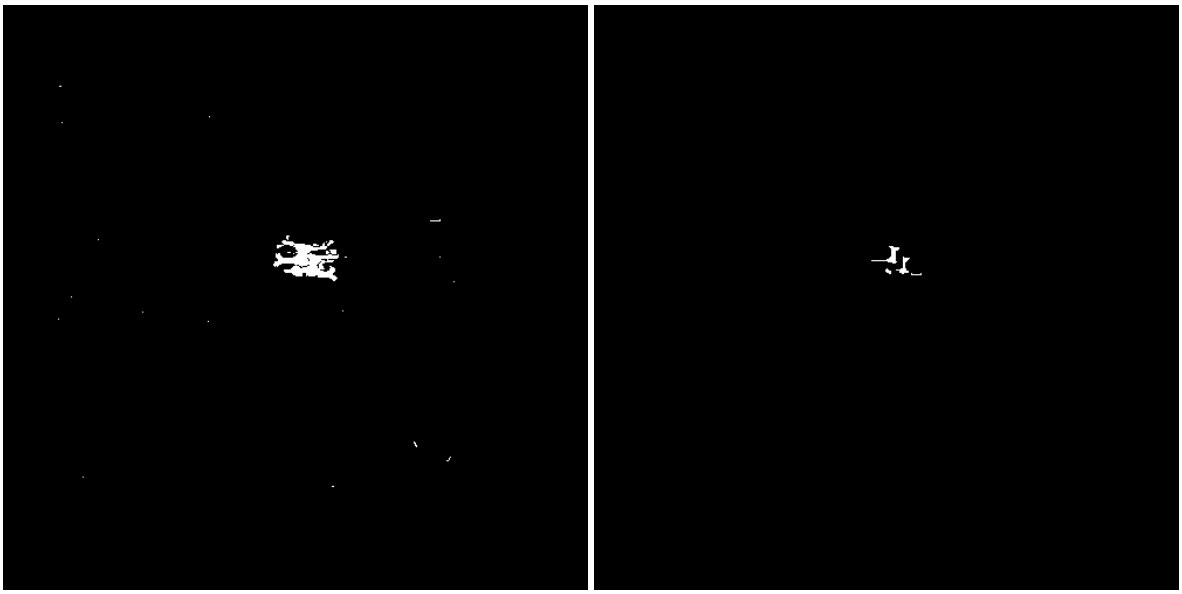
Abbildung 4.10: Beispiel einer Minkowski-Subtraktion

$$\begin{aligned} A &= \{(2,1);\ (3,1);\ (4,1);\ (2,2);(3,2);\ (4,2);\ (0,3);\ (1,3);(2,3);(3,3);\ (4,3);\ (5,3);\\ &\quad (0,4);\ (1,4);\ (2,4);\ (3,4);\ (4,4);\ (0,5);\ (1,5);\ (2,5);\ (3,5);\ (4,5)\};\\ B &= \{(0,0);\ (1,0);\ (0,1);\ (1,1)\}\\ A \ominus B &= \{(2,3);\ (3,3);\ (4,3);\ (2,4);\ (3,4)\} \end{aligned} \tag{4.8}$$

Hierbei ist A eine Struktur des Bildes und B das Strukturelement, dass komplementär zur Dilation auch in OpenCV eine Ellipse, ein Rechteck oder ein Kreuz sein kann. Erodieren bewirkt eine Verkleinerung der Struktur. Auch kann dadurch ein zusammenhängendes Objekt getrennt werden und Details wie z.B. Rauschpixel und kleine Strukturen, die kleiner als das Strukturelement sind, entfernt werden. Ein Beispiel einer Erosion mit einen Quadrat der Seitenlänge 5 und den Mittelpunkt P(2/2) ist in Bild 4.11 dargestellt.

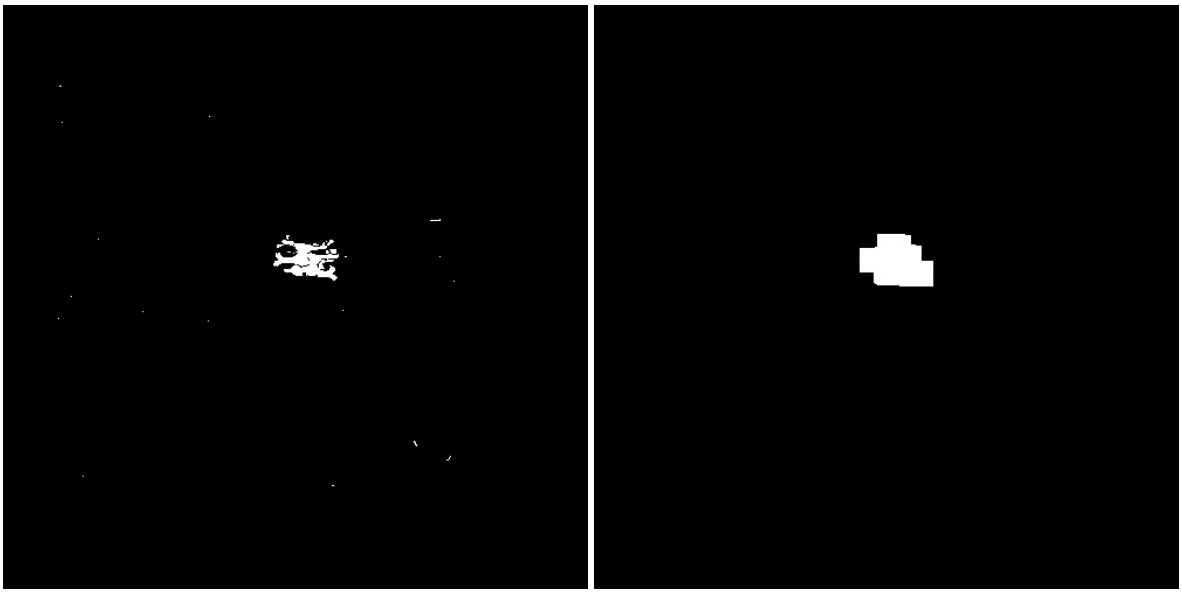
4.3.3 Opening

Das Opening besteht aus einer Erosion gefolgt von einer Dilation. Durch die Erosion werden zuerst kleine Elemente entfernt, durch die Dilation im Anschluss werden danach die übrig gebliebenen Strukturen wieder vergrößert, Löcher verkleinert und Strukturen wachsen zusammen. Diese Methode eignet sich sehr gut um erst Bildrauschen und kleine Objekte herauszufiltern und anschließend Objekte wieder zusammenzuführen, um so zusammengehörige Strukturen zu verbinden. Ein Opening ist in Abbildung 4.12 dargestellt.



(a) Threshold Bild (b) Bild nach Anwendung der Erosion

Abbildung 4.11: Beispiel für eine Erosion

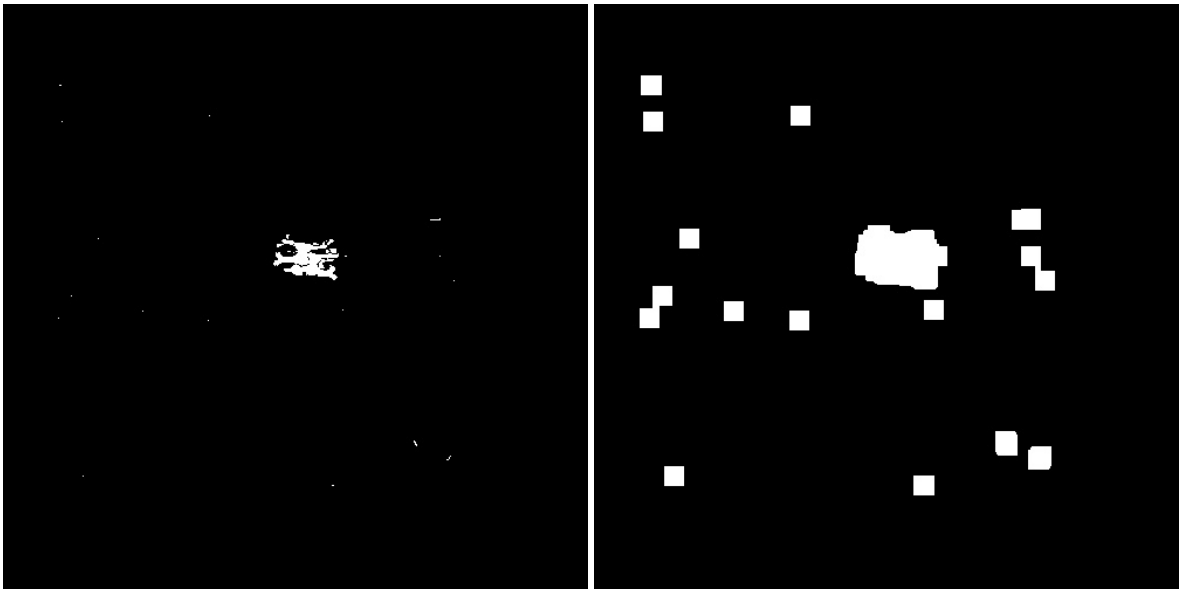


(a) Threshold Bild (b) Bild nach Anwendung des Opening

Abbildung 4.12: Beispiel für ein Opening

4.3.4 Closing

Das Closing besteht aus einer Dilation gefolgt von einer Erosion. Die Dilation bewirkt erneut eine Vergrößerung der Strukturen und Schließung der Löcher einer Struktur, die anschließende Erosion, bewirkt eine Verkleinerung der Segmente. Hierdurch können kleine Bildlücken und Risse geschlossen werden. Ein Closing ist in Abbildung 4.13 dargestellt.



(a) Threshold Bild (b) Bild nach Anwendung des Closing

Abbildung 4.13: Beispiel für ein Closing

4.3.5 Bildglättung

Bilder können weiterhin durch die Bildglättung verändert werden. Dies kann entweder durch einen Tiefpass- oder Hochpassfilter passieren. Während der Tiefpassfilter Bildrauschen entfernt, wird der Hochpassfilter für das Finden von Kanten in Bildern verwendet. Bildglättung wird durch Faltung des Bildes mit einen Kern durchgeführt. Da lediglich die Tiefpassfilterung für die Bildverarbeitung zum Filtern des Bildrauschens für die Zwecke des Programms interessant ist, werden im folgenden Kapitel verschiedene Möglichkeiten eines Tiefpasses für Bilder aufgezeigt. Da die Bildglättung hauptsächlich für Farbbilder geeignet ist, muss nach Ausführung des Algorithmus eine erneute Thresholdbildung stattfinden, da sonst die Bilder nicht mehr den Wert 0 oder 1 haben, sondern je nach ausgewählter Methode unterschiedliche Ergebnisse herauskommen können.

Mittelwert-Filter

Beim Mittelwertfilter wird das Bild mit einen normalisierten Boxfilter gefaltet. Hierbei wird einfach der Mittelwert aus allen Pixeln gebildet, die sich unter dem Kern befinden. Das zentrale Pixel wird dann mit dem Mittelwert ersetzt. Ein 3x3-Kern hat z.B. die Form: $\frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$

Median-Filter

Eine weitere Möglichkeit ist die Bildung des Medians mit Hilfe eines Kerns. Das zentrale Pixel wird dann mit dem Median-Wert ersetzt. Dieses Verfahren ist sehr effektiv gegen Salz- und Pfeffer-Rauschen, bei dem sich die Helligkeit des Rauschpixels stark von der Helligkeit der Nebenpixel unterscheidet.

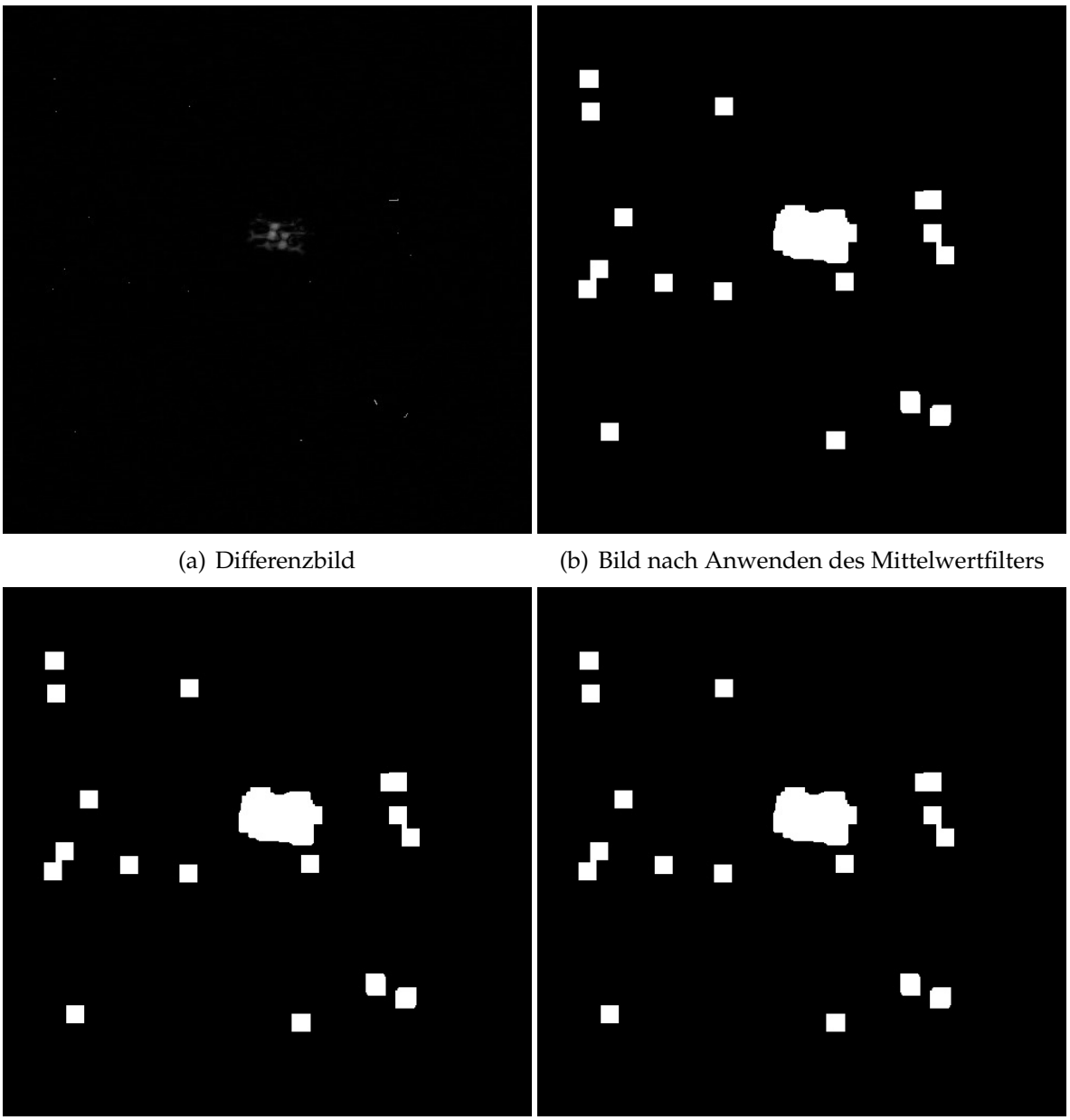
Gaußsche-Filter

Für die Anwendung des Gaußschen Filters wird ein sogenannter Gaußscher Kern verwendet. Hier übergibt man die Kernbreite und -höhe, sowie die Standartabweichung in X- und Y-Richtung. Die Gaußsche Filterung betrachtet sozusagen die Impulsantwort des Bildes. Hierdurch gehen kleinere Strukturen wie Bildrauschen verloren, gröbere Strukturen bleiben dagegen erhalten. Ein Beispiel für einen Gaußschen Kern ist in Folgenden gegeben: $\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix}$

Beim Betrachten der Bilder in Abbildung 4.14 fällt auf, dass sich die Bilder angewandt auf Schwarz-Weiß Bilder kaum unterscheiden und die Resultate sehr ähnlich sind. Jedoch wird bei allen Bildern stets eine Zusammenführung des Objekts vergleichbar mit dem Resultat aus 4.3.3 erreicht. Der Nachteil dieser Methode ist jedoch, dass auch das Bildrauschen weiterhin vorhanden ist.[ope18d, smo18]

4.4 Auswahl eines Algorithmen- und Parametersets

Um die geeignete Algorithmenkombination und ein geeignetes Parameterset der jeweiligen Algorithmen auszuwählen, wurde erst ein Video mit Hilfe des LadybugCapPro-Programms des Ladybug-SDKs aufgenommen. In diesen Video fliegt eine Parrot Bebop 2 Drohne startend vor der Kamera bis auf eine Entfernung von ungefähr 35 Meter weg und kehrt zum Startpunkt wieder zurück. Dieses Video wurde als Referenzvideo zur Bestimmung der geeigneten Algorithmen verwendet. Anschließend wurden auf dieses Video die Algorithmen mit verschiedenen Parameterwerten einzeln angewandt, um die Effekte auf das Bild zu testen. Hierbei wurde auch die Dauer der einzelnen Algorithmen im Mittel bestimmt. Hierzu ist eine Übersicht in Tabelle 4.1 zu sehen. Hier lässt sich erkennen, dass die verschiedenen Bildglättungsalgorithmen aus



(a) Differenzbild (b) Bild nach Anwenden des Mittelwertfilters

(c) Bild nach Anwendung des Median-Filters (d) Bild nach Anwenden des Gauß-Filters

Abbildung 4.14: Übersicht der verschiedenen Bildglättungsalgorithmen

Kapitel 4.3.5 mit 126,2 ms extrem langsam sind. Somit hat man sich für eine Erosion mit anschließender Dilation entschieden. Für die Erosion stellte sich eine Kerngröße von fünf als ideal heraus, da somit die Rauschpixel sehr gut entfernt werden können. Andere Einstellungen filterten entweder zu wenig Rauschpixel heraus oder höhere Einstellungen bewirkten einen Reichweitenverlust des Systems, da so zu viele Pixel der Drohne erodiert wurden. Für die Dilation stellte sich ein sehr großer Kerngröße von 31 als ideal heraus, da so die einzelnen Rotoren der Drohne in allen Entfernungen

zu einer Fläche verbunden wurden und somit die Drohne stets als Ganzes erkannt wurde. Bei Werten kleiner als 31 wurden gerade bei kurzer Entfernung der Drohne zur Kamera oft die einzelnen Rotoren als ROI angezeigt, während größere Werte des Kerns zusätzlichen Rechenaufwand bedeuteten. Weiterhin benötigte die Ausführung dieser beiden Algorithmen sehr viel Zeit, weswegen diese vorerst auf die Grafikkarte ausgelagert wurden. Hiermit konnte die Zeit der beiden Algorithmen bei der Dilation im Mittel um 20 ms verkürzt werden und bei der Erosion im Mittel um 14 ms. Hierbei wurden die Mittelwerte aus 100 Zeitmessungen betrachtet. Abschließend wurde die Idee umgesetzt, dass ein mehrfaches Ausführen der Algorithmen mit einen kleinen Kern ähnliche Effekte haben sollte, wie das einmalige Ausführen des Algorithmus mit einen großen Kern. Somit konnte zusätzlich ein Geschwindigkeitsgewinn bei der Erosion und Dilation erreicht. Als Parameterset hat sich somit die Kombination aus einer Erosion mit Kerngröße von 3 und einer vierzehnmaligen Dilation mit Kerngröße 3 als ideal herausgestellt.

Operation	Dauer in ms
Erosion	19,9 ms
Dilation	11,52 ms
Glättung	126,2 ms

Tabelle 4.1: Dauer der einzelnen Morphologischen Operationen

4.5 Konturenbestimmung

Nachdem das Bild nun vollständig segmentiert ist, muss als nächstes eine Erkennung der interessanten Regionen des Bildes erfolgen. Zur pixelgenauen Konturenerfassung wird die OpenCV-Funktion findContours() [fin18] von OpenCV verwendet. Diese verwendet den Algorithmus von Suzuki, S. and Abe [SA85]. Dieser Algorithmus tastet ein Binärbild systematisch ab und verfolgt Ränder. Die Funktion identifiziert miteinander verbundene Bereiche im Binärbild und gibt eine Liste aller identifizierten Konturen zurück. Als Ergebnis der Funktion erhält man die pixelgenauen Konturen mit einer Liste aus x- und y-Werten der zugehörigen Pixel in einer Baumstruktur zurück, die von der äußeren Kontur beginnend nach innen geordnet ist. Der Algorithmus unterscheidet zwischen Lochkonturen also schwarze Pixel im weißen Bereich und äußeren Konturen, also weiße Pixel im schwarzen Bereich. Ein Beispiel ist in Abbildung 4.15 dargestellt. Hier ist auch die Unterscheidung zwischen äußeren Konturen und Lochkonturen gut zu erkennen. Die Grenzen im Bild, an denen ein Übergang erkannt wird, sind mit einen B markiert, die zusammengehörigen Regionen mit einen S.[fin18, SA85]

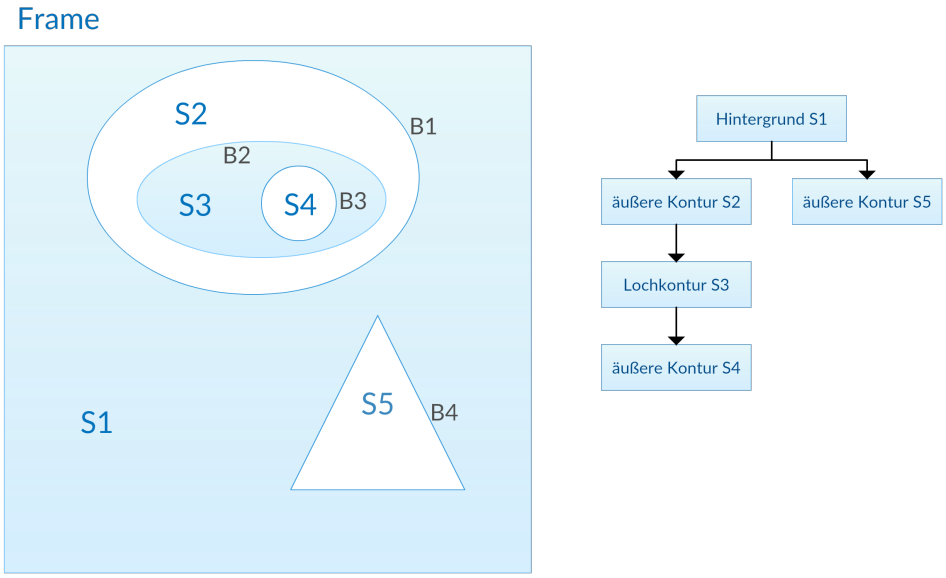


Abbildung 4.15: Beispielbild des Konturenalgorithmus nach Suzuki, S. and Abe[SA85]

4.6 Schwerpunkt- und Winkelbestimmung

Nachdem die Konturen und Flächen der Objekte, die als Detektion zählen, gefunden wurden, muss als nächster Schritt die Berechnung des Schwerpunktes der einzelnen Regionen erfolgen und aus diesen Regionen schließlich die Winkel Azimuth und Elevation der einzelnen Objekte berechnet werden. Zuerst erfolgt die Bestimmung der Schwerpunkte der Objekte. Dies wird umgesetzt, indem x_{min}, x_{max}, y_{min} und y_{max} bestimmt werden. Schließlich wird ein Rechteck der Breite x_{max}-x_{min} und der Höhe y_{max}-y_{min} mit den Startpunkt P (x_{min}/y_{min}) erstellt. Nachdem man dieses umschließende Rechteck erstellt hat, wird das Schwerpunkt-Pixel einfach berechnet durch:

$$x_{SP} = \frac{x_{max} + x_{min}}{2} \tag{4.9}$$

$$y_{SP} = \frac{y_{max} + y_{min}}{2} \tag{4.10}$$

$$SP\ (x_{SP}/y_{SP}) \tag{4.11}$$

Als nächstes erfolgt die Berechnung der Winkelkoordinaten SP(λ_{SP}/ϕ_{SP}) der einzelnen ROIs aus den Pixelkoordinaten der Schwerpunkte SP(x_{SP}/y_{SP}) relativ zum Ursprung des Bildes. Hierbei wird davon ausgegangen, dass das Panoramabild wie eine Art Kreis um die Kamera liegt. Dieser Ursprung 0 liegt immer in der Mitte des Bildes und hat die Koordinaten:

$$x_O = \frac{bildweite}{2} \tag{4.12}$$

$$y_O = \frac{bildhoehe}{2} \tag{4.13}$$

$$O\,(x_O/y_O) \tag{4.14}$$

Der Breitenwinkel (Azimut) λ und der Höhenwinkel (Elevation) ϕ ergeben sich dann aus den Kamerabild wie folgt:

$$\lambda_{rel} = \begin{cases} \frac{x_{SP}-x_O}{bildweite} * 2\pi \text{ für } x_{SP} \geq x_O \\ 2 * \pi - \frac{-x_{SP}+x_O}{bildweite} * 2\pi \text{ für } x_{SP} < x_O \end{cases} \tag{4.15}$$

$$\phi_{rel} = \frac{-y_{SP} + y_O}{bildhoehe} * \pi \tag{4.16}$$

Die Darstellung der Winkel ist in Abbildung 4.16 zu sehen.

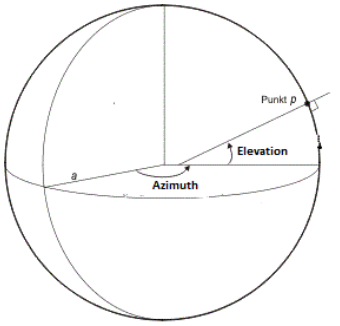


Abbildung 4.16: Skizze des Breiten und Höhenwinkels

Das erkannte Objekt mit eingezeichneten Schwerpunkt und umschließenden Rechteck ist in Abbildung 4.17 zu sehen.

4.7 Objektverfolgung

Um eine grundlegende Objektverfolgung zu garantieren, musste ein Algorithmus programmiert werden, der es ermöglicht, zwischen den Detektionen eine Zuordnung durchzuführen, die es den Benutzer erleichtert, zwischen Bildern die selbe Drohne zu verfolgen. Da der Benutzer die Weitergabe der Winkel des gewünschten Zielobjekts über die GUI mittels ComboBoxen steuert, muss hier eine Zuordnung der Drohnen in das jeweilige Fenster erfolgen, da sonst unterschiedliche Winkel von Iteration zu Iteration ausgegeben werden würden. Laut Kapitel 2.2.1 ist die Anforderung 1300 "Verfolgung von mindestens drei Drohnen mit passender Zuordnung zwischen den Bildern" ein Kann-Kriterium ist, wurde diese Verfolgung relativ simpel umgesetzt. Eine Beispielausgabe des Objektverfolgungsalgorithmus ist in Beispiel 1 zu sehen. Das

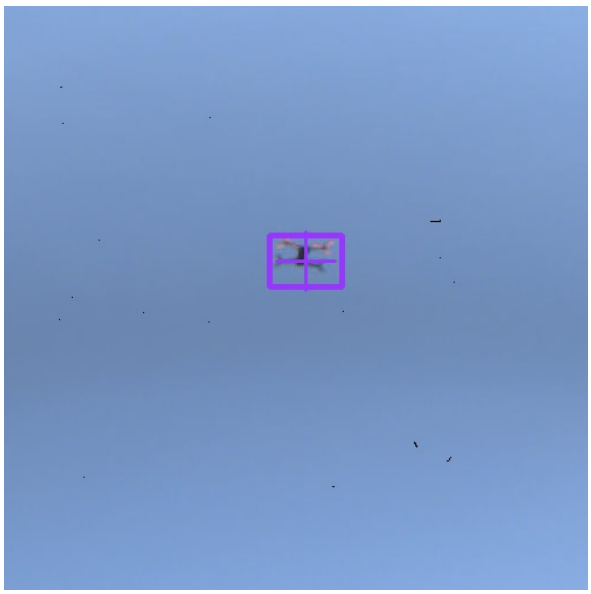


Abbildung 4.17: Farbbild am Ende der Verarbeitungskette mit eingezeichneter ROI

Softwareablaufdiagramm ist in Abbildung 4.18 zu sehen. Hierfür wurde als erstes eine Sortierung der erkannten Konturen nach Größe vorgenommen, d.h. es werden lediglich immer die maximal fünf größten Konturen miteinander verglichen, die anderen werden ignoriert. In Beispiel 1 sind die letzten zwei Detektionen in Schritt 1 dargestellt. Anschließend wurden die Winkelabstände der beiden Detektionslisten durch die Anwendung des Seitenkosinussatz der sphärischen Geometrie[ent18] zueinander berechnet, was in Schritt 2 zu sehen ist (vergl. Abbildung 4.19). Somit berechnet sich der Abstand d zwischen zwei Detektionen mit den Schwerpunkten $S_1(\lambda_1/\phi_1)$ und $S_2(\lambda_2/\phi_2)$ wie folgt:

$$d = \arccos(\sin(\lambda_1)\sin(\lambda_2) + \cos(\lambda_1)\cos(\lambda_2)\cos(\phi_2 - \phi_1)) \tag{4.17}$$

Nach diesem Schritt erhält man zwischen zwei Iterationsschritten bis zu 25 Abstände der maximal jeweils 5 Detektionen. Anschließend werden diese 25 Abstände beginnend mit dem kleinsten aufsteigend sortiert (vergl. Schritt 3 Beispiel 1). Anschließend werden die fünf kleinsten Abstände, bei denen der aktuelle Index der Detektion und der vorherigen Detektion nicht benutzt wird, in die Liste eingefügt, solange der Abstand kleiner als $\pi/6$ ist, da davon ausgegangen wird, dass es sich bei einen größeren Abstand um ein neues Objekt handelt und nicht um das selbe Objekt, das durch den Algorithmus zugeordnet wurde(vergl. Schritt 4 Beispiel 1). Abschließend werden noch Detektionen, die als neu angesehen werden, in die Liste mit aufgenommen und diese Liste schließlich in der GUI publiziert(vergl. Schritt 5 und Schritt 6 Beispiel 1). Somit konnte ein Algorithmus entwickelt werden, der sowohl auch neue Detektionen berücksichtigt, falls die Anzahl der Detektionen im aktuellen Schritt höher ist als die Anzahl der Detektionen im vorherigen Schritt, als auch alte Detektionen löscht, falls die Anzahl der aktuellen Detektionen höher ist, als die Anzahl der Detektionen

im vorherigen Schritt. Weiterhin konnte so eine erste und einfache Zuordnung von Detektionen zweier aufeinanderfolgenden Iterationen erreicht werden, was die Verfolgung und Winkelausgabe der erkannten Objekte in der GUI deutlich erleichtert.

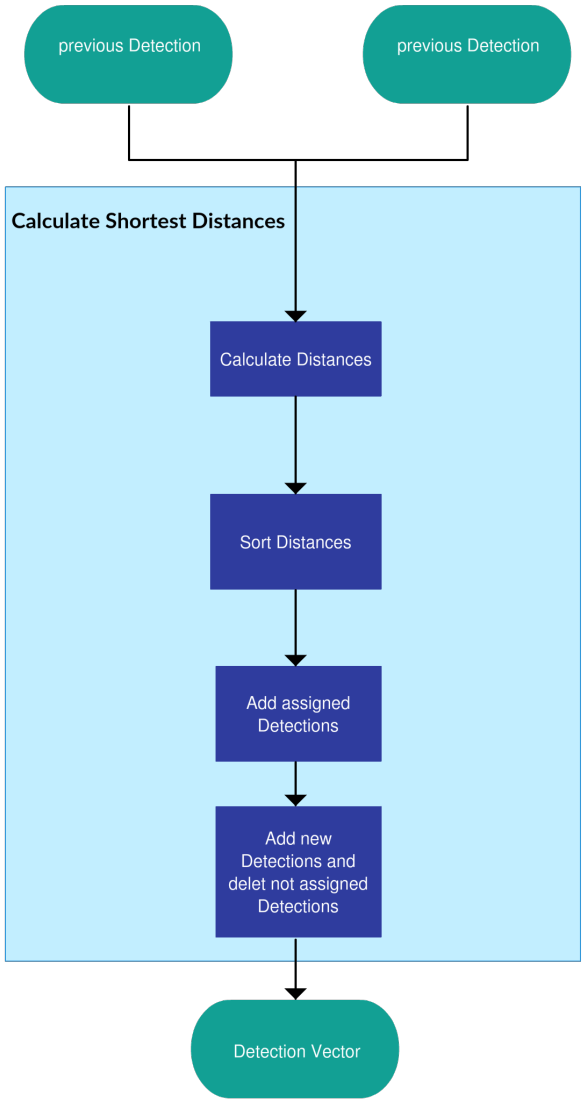


Abbildung 4.18: Ablaufdiagramm des Objektverfolgungsalgorithmus

Schritt 1:

Current Detection:
Azimuth Elevation
0.310669 0.915716
0.253655 0.898263
3.067130 0.051196

Previous Detection:
Azimuth Elevation
0.254818 0.906408
0.309505 0.913389

Schritt 2:

All distance combinations:
previndex: 0 currentindex: 0 Distance: 0.03547
previndex: 0 currentindex: 1 Distance: 0.00817
previndex: 0 currentindex: 2 Distance: 2.14409
previndex: 1 currentindex: 0 Distance: 0.00243
previndex: 1 currentindex: 1 Distance: 0.03763
previndex: 1 currentindex: 2 Distance: 2.12389

Schritt 3:

Sort final by distances:
previndex: 1 currentindex: 0 Distance: 0.00243
previndex: 0 currentindex: 1 Distance: 0.00817
previndex: 0 currentindex: 0 Distance: 0.03547
previndex: 1 currentindex: 1 Distance: 0.03763
previndex: 1 currentindex: 2 Distance: 2.12389
previndex: 0 currentindex: 2 Distance: 2.14409

Schritt 4:

Sorted final by Current Index:
previndex: 1 currentindex: 0 Distance: 0.00243
previndex: 0 currentindex: 1 Distance: 0.00817

Schritt 5:

Adding new Detections
Adding index as new objekt 2 at vector position: 2

Schritt 6:

Final Point Vector with size 5 and currentDetections 3:
Azimuth Elevation
0.253655 0.898263
0.310669 0.915716
3.067130 0.051196

Beispiel 1: Beispielausgabe des Objektverfolgungsalgorithmus

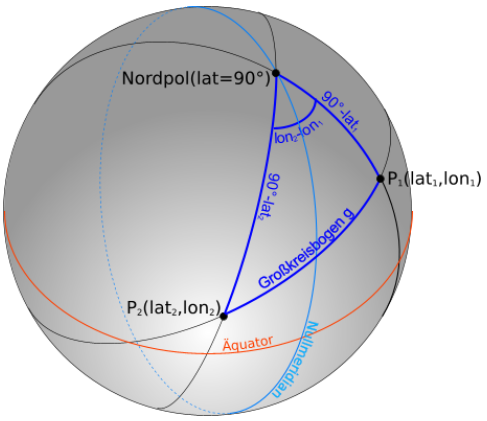


Abbildung 4.19: Beispielskizze zur Berechnung von Winkelabständen aus Längen- und Breitengraden[ent18]

5 Programmaufbau

Im folgenden Kapitel wird der Aufbau und die Funktionsweise der Software beschrieben. Die Abbildungen dieses Kapitels sind auf Grund ihrer Größe am Ende des Kapitels zu sehen. Eine Klassenübersichtsdiagramm ist in Abbildung 5.1 zu sehen. Dieses Programm zeigt, wie die einzelnen Klassen miteinander zusammenhängen. Hierbei wurden die Klassen dem Programmablauf nach von links nach rechts angeordnet.

5.1 Wichtige Klassen des Programms

Um das Programm objektorientiert programmieren zu können, wurden mehrere Klassenstrukturen definiert, die die Umsetzung erleichtern und unterstützen sollen. Die wichtigste Klasse „Image“ hat als Attribute die Aufnahmezeit des Bildes und das Farbbild in Form einer OpenCv Matrix. Die Klasse ProcessedImage erbt die Attribute der Image Klasse und hat als zusätzliches Attribut eine OpenCV Matrix, in der das Thresholdbild abgelegt ist. In dieser Form werden auch die fertig bearbeiteten Bilder mit markierten ROIs gespeichert. Die Klasse „Detection“ hat zwei Attribute, eine Liste mit den Azimut- und Elevationswinkeln der fünf größten Schwerpunkte und eine Liste mit den zugehörigen Thumbnails, die die Objekte in Nahaufnahme zeigen. Ein Detektionsobjekt wird immer nach dem Erstellen des Differenz- und Thresholdbildes erstellt, wenn die Software nach den ROIs sucht. Anschließend werden die Detektionen im „Detection Logic Thread“ miteinander verglichen um sie einander zuordnen zu können. Für die Zuordnungen wird die Klasse „Distances“ benötigt. Diese Klasse hat drei Attribute. Einerseits speichert sie die Positionindizes der Detection-Punkte in der Detection-Liste der beiden Detections, die miteinander verglichen werden. Andererseits wird die Distanz als Double zwischen den beiden Punkten gespeichert. Diese Klasse arbeitet als Unterstützungsklasse, um bei zwei aufeinanderfolgenden Detektionen die Schwerpunkte einander zuzuordnen, um so die Drohnen verfolgen zu können.

5.2 Übersicht

Der Programmaufbau ist in Abbildung 5.2 zu sehen. Die Software besteht grundlegend aus vier Threads, die einer Bildverarbeitungskette entsprechen. Zwischen diesen Threads ist jeweils ein Ringbuffer plaziert, in dem die jeweiligen Bilddaten nach ihrem Verarbeitungsschritt abgespeichert werden, bis der nachfolgende Thread bereit

ist den darauffolgenden Schritt durchzuführen. Somit soll eine unabhängige, möglichst schnelle und parallele Prozessierung der Threads ermöglicht werden, um den Geschwindigkeitsanforderungen des Systems zu genügen. Die Arbeitsaufteilung der Threads wurde bestimmt durch die Messung der Prozessierungsdauer der einzelnen Schritte und das Aufteilen, so dass alle Threads möglichst gleich lange dauern und die Latenzzeiten somit möglichst gering sind. Die durchschnittliche Dauer der einzelnen Threads bei 100 Messungen ist in Tabelle 5.1 zu sehen. Die Aufgaben der einzelnen Threads sowie die Funktion des Ringbuffers werden im Folgenden beschrieben.

Threadname	**Dauer**
Ladybug Capture Thread	212,38 ms
Image Threshold Thread	249,1 ms
Image Processing Thread	168,84 ms
Detection Logic Thread	10,98 ms

Tabelle 5.1: Durchschnittliche Thread-Dauer der einzelnen Threads bei 100 Messungen

5.2.1 Ladybug Capture Thread

Das Softwareablauf-Diagramm ist in Abbildung 5.3 dargestellt. Der Ladybug Capture Thread ist, je nach Einstellung, entweder für das Einlesen eines Videos oder für das Aufnehmen der Bilder der Kamera zuständig. Im Modus des Videoeinlesens wird als Erstes ein Fenster zur Auswahl eines Videos geöffnet. Der Thread initialisiert dann ein OpenCV VideoCapture Objekt (vergl. [ope18c]). Anschließend wird das Video Bild für Bild eingelesen und im Image-RingBuffer abgespeichert.

Im Kamerabetriebsmodus wird ein neuer LadybugContext erstellt und anschließend die Kamera gestartet. Nach der Initialisierung der Kamera wird beim ersten Aufruf ein erstes Panoramabild im Programmordner gespeichert, das zur Erstellung eines Maskenbildes benutzt werden kann. Danach werden in einer Endlos-Schleife Bilder aufgenommen, zu einem Panoramabild gestitcht, das Panoramabild in eine OpenCV Matrix umgewandelt und anschließend im Image-RingBuffer abgespeichert. Um ein Panoramabild aufzunehmen, werden stets zuerst sechs Einzelbilder aufgenommen. Diese werden anschließend transformiert und letztendlich auf der Grafikkarte zu einem Panoramabild gestitcht.

5.2.2 Image Threshold Thread

Für die Erstellung eines Thresholdbildes wurde ein separater Thread gewählt, da die Erstellung des Bildes aufgrund der Größe der Panoramabilder relativ viel Zeit in Anspruch nimmt, wie in Tabelle 5.1 zu sehen ist. Der Ablauf des Threads ist in Abbildung 5.4 dargestellt. Dieser Thread arbeitet stets mit zwei aufeinanderfolgenden

Bildern. Das letzte Bild wird previousImage genannt, das aktuelle Bild currentImage. Zu Beginn eines Aufrufs überschreibt der Thread stets das letzte Bild mit dem aktuellen Bild. Lediglich beim ersten Aufruf ist nur ein Bild vorhanden, deswegen wird nur das aktuelle Bild abgelegt. Anschließend überprüft der Thread, ob ein Masken-Bild geladen wurde, welches Bereiche abdeckt, die nicht in der Detektion betrachtet werden sollen. Ist dies der Fall, so legt der Thread das Maskenbild über das aktuelle Bild. Anschließend erstellt der Thread aus dem letzten und dem aktuellen Bild ein Thresholdbild, wie in Kapitel 4.2 beschrieben. Dies wird durch die Umwandlung der beiden Bilder in Grauwertbilder, die Differenzbildbildung und abschließender Thresholdbildung durchgeführt. Abschließend wird ein neues Processed Image Objekt, das aus dem aktuellen Bild, dem Thresholdbild und der Aufnahmezeit besteht, erstellt und im ersten Processed Image RingBuffer 1 abgespeichert.

5.2.3 Image Processing Thread

Der Image Processing Thread ist für die eigentliche Bildverarbeitung zuständig. Hierzu lädt er zuerst ein Processed Image Objekt aus dem ersten Processed Image Buffer 1. Anschließend übergibt er das Farbbild und Thresholdbild an ein Image Processing Objekt, welches die einzelnen Bildverarbeitungsfunktionen, die in 4.3 beschrieben werden, umsetzt. Der genaue Ablauf dieser Methode wird in Kapitel 5.2.4 erläutert. Die Rückgabe dieser Funktion besteht einerseits aus dem Farbbild mit den zugehörigen Markierungen der ROI in diesem Bild sowie dem Thresholdbild, die im zweiten Processed Image RingBuffer 2 abgelegt werden. Andererseits wird eine Liste der gefundenen Detectionen mit zugehörigen Schwerpunkt und Thumbnails als Liste, die im sogenannten Detection RingBuffer abgespeichert werden, zurückgegeben.

5.2.4 Image Processing Klasse

Die Image Processing Klasse verwirklicht alle Image Processing Funktionen. Sie hat für die verschiedenen Image Processing Funktionen, die in Kapitel 4.3 beschrieben sind, die nötigen Parameter als Attribute gespeichert. Diese Attribute können jeweils über die GUI verändert werden. Dadurch kann das Programm während des Ablaufs verändert werden. Die wichtigste Methode ist die Funktion „searchForMovement“, die vom Image Processing Thread aufgerufen wird. Diese hat als Eingangsparameter ein Thresholdbild und ein zugehöriges Farbbild. Zuerst werden die Bildverarbeitungsfunktionen auf das Thresholdbild angewandt. Hierbei können einzelne Funktionen jeweils über einen Boolean und zugehöriger CheckBox (vergl. [qch18]) zu- und abgeschaltet werden. Anschließend wird nach interessanten Regionen, also Regionen in denen das Thresholdbild weiß ist, mittels der findCountours-Funktion von OpenCV gesucht (vergl. Kapitel 4.5 und [fin18]). Danach werden diese Konturen der Größe nach absteigend geordnet, da nur die fünf größten Konturen betrachtet werden sollen. Abschließend erfolgt eine Schwerpunktbestimmung für jede Kontur mittels OpenCV Bounding Rectangle (vergl. [fin18]), wodurch dann der Azimuth- und Elevationswin-

kel mittels Berechnung wie in Kapitel 4.6 bestimmt werden kann. Hierbei wird für jedes Bild ein Detection Objekt erstellt, das aus einer Liste der Schwerpunkte und einer Liste der zugehörigen Thumbnails besteht. Die Funktion gibt schließlich das Farbbild mit markierten ROIs, das Thresholdbild nach der Veränderung durch die Bildverarbeitungsfunktionen und das Detektions-Objekt mit zugehörigen Winkeln und Thumbnails an den Image Processing Thread zurück. Der Ablauf der Image Processing searchForMovement-Methode ist in Abbildung 5.6 dargestellt.

5.2.5 Detection Logic Thread

Der Detection Logic Thread ist der letzte Thread des Programms. Dessen Ablaufdiagramm ist in Abbildung 5.7 dargestellt. Dieser Thread überspeichert als Erstes die letzte Detection, prevDetection genannt, mit der aktuellen Detection, currentDetection genannt. Anschließend lädt er sowohl eine neue Detection in die currentDetection, als auch das zugehörige Processed Image. Weiterhin wird überprüft, ob im aktuellen Bild Detections gefunden wurden. Wurden keine gefunden, so wird lediglich das Threshold und das Farbbild an die GUI weiter gegeben, falls die Iteration veröffentlicht werden soll. Wurde eine Detection gefunden, dann kommt der Objektverfolgungsalgorithmus aus Kapitel 4.7 zum Einsatz. Anschließend wird überprüft, ob die Winkel der Detection via TCP/IP-Schnittstelle (vergl. Kapitel 3.7) gesendet werden sollen. Ist dies der Fall, so wird ein QSignal (vergl. [qsi18]) mit den zugehörigen Detektionen ausgesendet. Abschließend wird ebenfalls überprüft, ob das Farbbild, das Thresholdbild und die zugehörigen ROIs an die GUI übermittelt werden sollen, um die Bilder und die passenden Winkel dazu anzuzeigen. Dies erfolgt ebenfalls über ein QSignal.

5.3 Übergabe der Bilder von Thread zu Thread

Da die zuvor beschriebenen Threads parallel laufen, aber jeweils für einander Daten produzieren und konsumieren, werden Zwischenspeicher benötigt, die die Bilder zwischenspeichern bevor sie weiterverarbeitet werden. Hierfür wurden RingBuffer als Datenstruktur implementiert. Ein RingBuffer besteht aus einen Array von Objekten, und funktioniert nach dem First-In-First-Out Prinzip, d.h. die Objekte werden am Schwanz eingefügt und werden am Kopf des Buffers entnommen. Erreicht der Kopf oder der Schwanz das Ende des Arrays, so springt er wieder auf die erste Stelle des Arrays und führt seine Arbeit fort. Die Funktionsweise des RingBuffers ist in Abbildung 5.8 zu sehen. Da der Producer und der Consumer-Thread unterschiedliche Geschwindigkeit aufweisen, kann es schnell zu Dateiinkonsistenzen kommen. Wenn der Konsumer-Thread schneller als der Producer-Thread ist, so kann es sein, dass der Konsumer-Thread ein Objekt lädt, dass noch Null ist, oder ein altes Bild lädt. Ist der Producer-Thread schneller als der Konsumer-Thread, dann besteht die Gefahr, dass der Producer-Thread noch nicht abgearbeitete Dateien überschreibt. Im schlimmsten Fall greifen beide Threads im selben Moment auf das gleiche Element zu, wodurch es zu unvorhersehbaren Dateiinkonsistenzen kommen kann. Qt stellt

zwei Objekte bereit, um die vorherig genannten Fehler zu verhindern. QSemaphore bestehen aus einen Ressourcenzähler, den Methoden reservieren und freigeben. Für die Dateiverwaltung wurden hier zwei Semaphore benutzt, freeBytes-Semaphore und usedBytes-Semaphore. Der freeBytes Ressourcenzähler wird zu Beginn auf die maximale Anzahl der verfügbaren Ressourcen gesetzt, der usedBytes auf 0. Fügt der Producer nun eine Ressource in den RingBuffer ein, so wird der Ressourcenzähler von usedBytes um eins erhöht und der von freeBytes um 1 erniedrigt. Umgekehrt ist es, wenn der Konsumer eine Ressource konsumiert. Da die Ressourcen nie unter 0 und über die maximale RingBufferlänge gehen dürfen, werden Produzer und Konsumer gezwungen zu warten, bis wieder eine Ressource freigegeben ist. Somit kann das Überschreiben oder das Anfordern von alten oder leeren Daten verhindert werden. Als zusätzliche Absicherung hat der RingBuffer ein QMutex-Objekt. Mutexe verhindern durch gegenseitigen Ausschluss, dass nebenläufige Prozesse gleichzeitig oder zeitlich verschränkt gemeinsam genutzte Datenstrukturen unkoordiniert verändern, wodurch die Datenstrukturen in einen inkonsistenten Zustand geraten können. Somit kann das oben genannte Problem der Dateninkonsistenz gelöst werden. Der Mutex wird jeweils beim Speichern und Laden gelockt, so dass der andere Thread in dieser Zeit warten muss.[qse18, qmu18]

5.4 GUI

Um den Benutzer die Bedienung des Programmes zu erleichtern und um eine Präsentationsmöglichkeit des Programms für Messen und anderen Ausstellungen zu ermöglichen, wurde in den Zielen in Kapitel 2 das Umsetzen einer Grafischen Oberfläche mit Hilfe von Qt festgelegt. In folgendem Kapitel werden die Bedienelemente der Oberfläche beschrieben und die Reaktion des Programms auf User Input dargestellt.

5.4.1 Grafische Oberfläche des Programms

In diese, Kapitel werden der Aufbau und die verschiedenen Funktionen der GUI-Elemente beschrieben. Ein Bildschirmfoto der GUI ist in Abbildung 6.10 zu sehen. Die wichtigsten Elemente und deren Funktionsweise werden im Folgenden beschrieben. Die Dokumentation der Elemente ist in [qwi18] zu finden.

1. Anzeige des Farbbildes mit eingezeichneten Thumbnails und Anzeige des Maskenbildes

2. Anzeige des Thresholdbildes

3. Anzeige der einzelnen Thumbnails inklusive Azimut und Elevationsanzeige sowie Auswahl des zu verfolgenden Thumbnails mittels QRadioButtons

4. Aktivierung und Deaktivierung einer festen Thumbnail-Größe mittels QCheck-Box sowie die Einstellung der Länge und Breite der festen Thumbnail-Größe mittels QSpinBox

5. Einstellen des Threshold mittels QSlider

6. Aktivierung und Deaktivierung des Minimumfilters mittels QCheckBox und Veränderung der Anzahl der Pixel, die für eine Detektion nötig sind, mittels QSlider

7. Einstellung des Faktors, um den das Bild geschrumpft werden soll, mittels UISLider

8. Einstellung, nach wie vielen abgearbeiteten Frames die Bilderanzeigen der GUI aktualisiert werden sollen mittels QSlider

9. Aktivierung und Deaktivierung des Blur-Algorithmus mittels QCheckBox, Änderung der Kernel-Größe mittels QSlider und Änderung der verschiedenen Algorithmusvarianten mittels QComboBox

10. Aktivierung und Deaktivierung des Erodier-Algorithmus mittels QCheckBox, Änderung der Kernel-Größe mittels QSlider und Änderung der Kernel-Form mittels QComboBox

11. Aktivierung und Deaktivierung des Dilation-Algorithmus mittels QCheckBox, Änderung der Kernel-Größe mittels QSlider und Änderung der Kernel-Form mittels QComboBox

12. Start- und Quit-Button zum Starten und Beenden des Programms, sowie Buttons zum Sichern und Laden von Settings und dem Laden eines Videopfades oder Pfades eines Maskenbildes

13. QCheckbox für das Aktivieren des Kamera-Modus, das Speichern der Winkeldaten in einer Textdatei, das Senden der Daten via TCP/IP und ob Bilder der Kamera verworfen werden sollen, wenn der ImageBuffer voll ist

14. Aktivieren und Deaktivieren, ob nur eine bestimmte Region des Bildes betrachtet werden soll mittels QCheckBox, und Bestimmen der Startkoordinaten und Länge und Breite des Vierecks, das betrachtet werden soll mittels QSpinBox

15. Anzeige der verschiedenen FPS-Raten der einzelnen Threads

5.4.2 Reaktion des Programms auf Änderungen des Users

Jede grafische Oberfläche, die mit Hilfe von Qt implementiert wurde, verwendet Signals und Slots, um auf gewisse Änderungen zu reagieren. Alle Standart QWidgets,

d.h. alle Standart GUI Elemente, die in diesem Programm verwendet wurden, beinhalten bestimmte vordefinierte Signals, die ausgesandt werden, wenn z.B. eine Person auf einen Button drückt. Die Funktionsweise der Signal-Slot-Verbindungs ist in Abbildung 5.10 dargestellt. Zugehörig zum jeweiligen Signal wird ein Slot eines Objekts zugeordnet, welcher beschreibt, wie das Objekt auf die zugehörige Änderung zu reagieren hat. Hier gibt es für viele QWidget-Elemente auch zugehörige Slots, jedoch können auch eigene Slots für eigene Objekte oder QObjects implementiert werden. Die Vorteile des Signal-Slot-Mechanismus sind, dass dieser typensicher ist, d.h. die Signatur eines Slots muss der eines Signals entsprechen. Weiterhin sind Slots lose gekuppelt, was bedeutet, eine Klasse, die ein Signal emittiert, weiß nicht welcher Slot auf die Klasse reagiert. Jedoch wird durch die Verbindung des Signal-Slot-Mechanismus sichergestellt, dass die Slot-Funktion zur richtigen Zeit mit den richtigen Parametern ausgeführt wird. Durch diese Vorteile, kann sichergestellt werden, dass durch Signal-Slot-Verbindungen die jeweilig zugeordneten Objekte auf die User-Eingaben direkt reagieren und somit das Programm möglichst schnell auf veränderte Parametereingaben reagieren kann, ohne das Programm anhalten zu müssen. Ein Beispiel hierfür ist, wenn der User den Quit-Button drückt, so wird ein Signal buttonClicked vom Quit-Button ausgesandt, der zugehörige Slot verändert in allen Threads die doStop-Booleans, wodurch alle Threads beendet werden und das Programm schließt.[qsi18]

Die grafischen Oberflächenelemente wurden mittels QtCreator erstellt und definiert. Diese Elemente werden mittels XML-File im zugehörigen Programm geladen. Hierfür ist die Klasse „ladybugDroneTracker360GUIProgram“ zuständig. In dieser Klasse erfolgt die Organisation und Steuerung des kompletten Ablaufs aller Grafischen Oberflächenelemente. Hierfür werden nach dem Laden des XML-Files alle zugehörigen Signal-Slot-Beziehungen definiert. weiterhin sind hier die onAction()-Funktionen der einzelnen GUI-Elemente definiert, d.h. was passieren soll, nachdem ein bestimmtes GUI-Element betätigt worden ist. Somit ist diese Klasse die Schnittstelle zwischen Benutzer und der Programmlogik im Hintergrund, weil von dieser alle Attribute der Bildverarbeitungsalgorithmen verändert werden. Auch werden von hier die einzelnen Threads gestartet und beendet und die grafische Oberfläche aktualisiert und gesteuert.

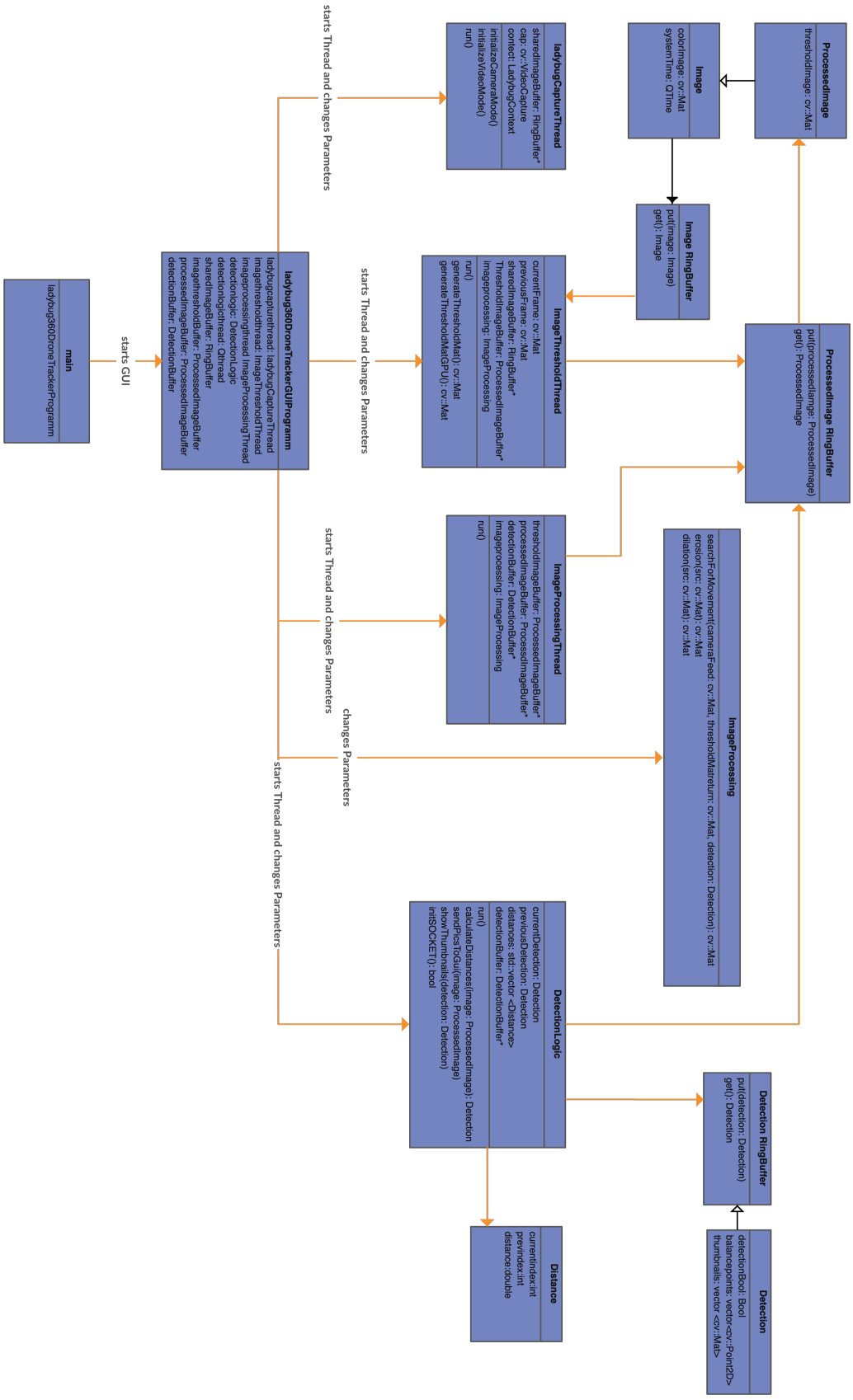


Abbildung 5.1: Klassenübersichtsdiagramm des Programms

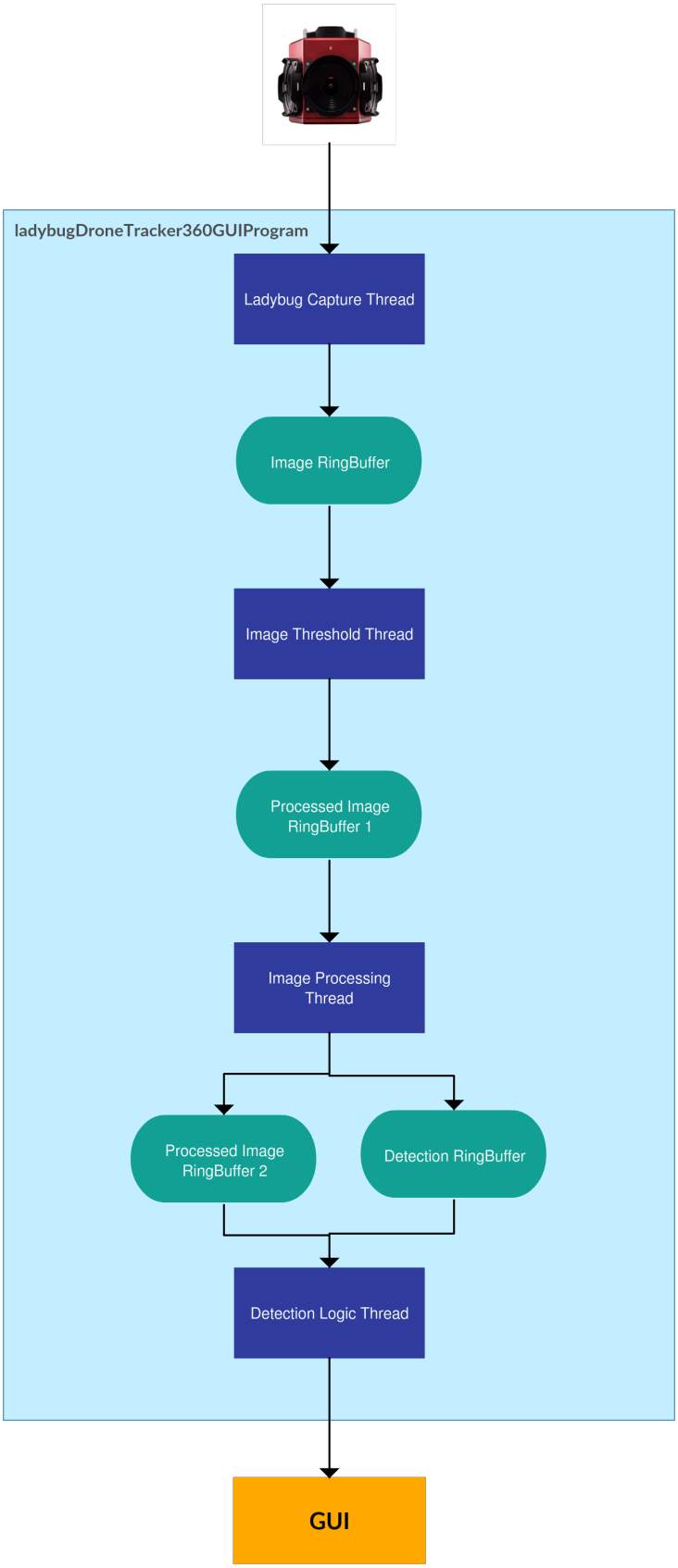


Abbildung 5.2: Softwareflussdiagramm des Programms[lad18b]

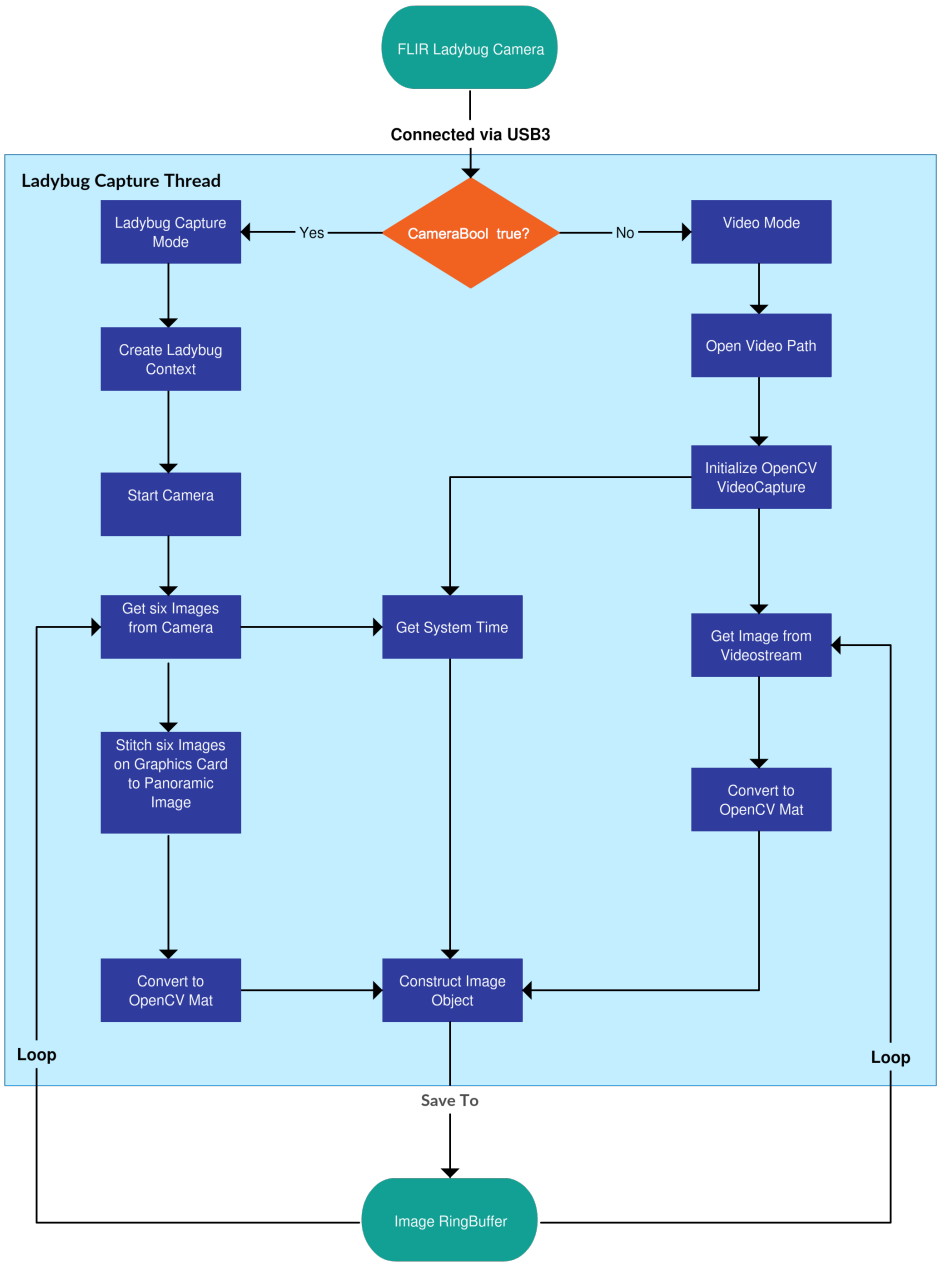


Abbildung 5.3: Softwareablaufdiagramm des Ladybug Capture Threads

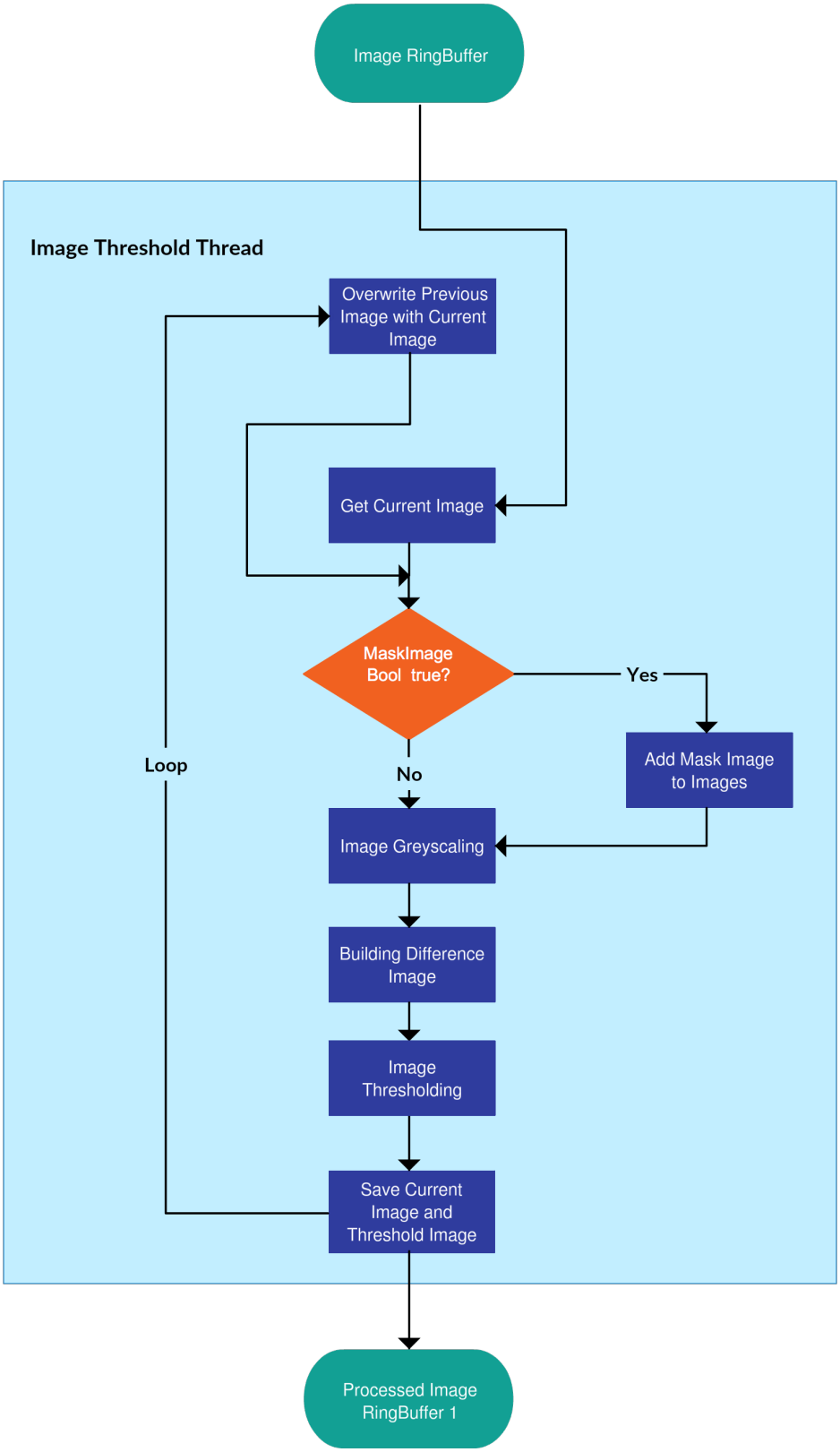


Abbildung 5.4: Softwareablaufdiagramm des Image Threshold Threads

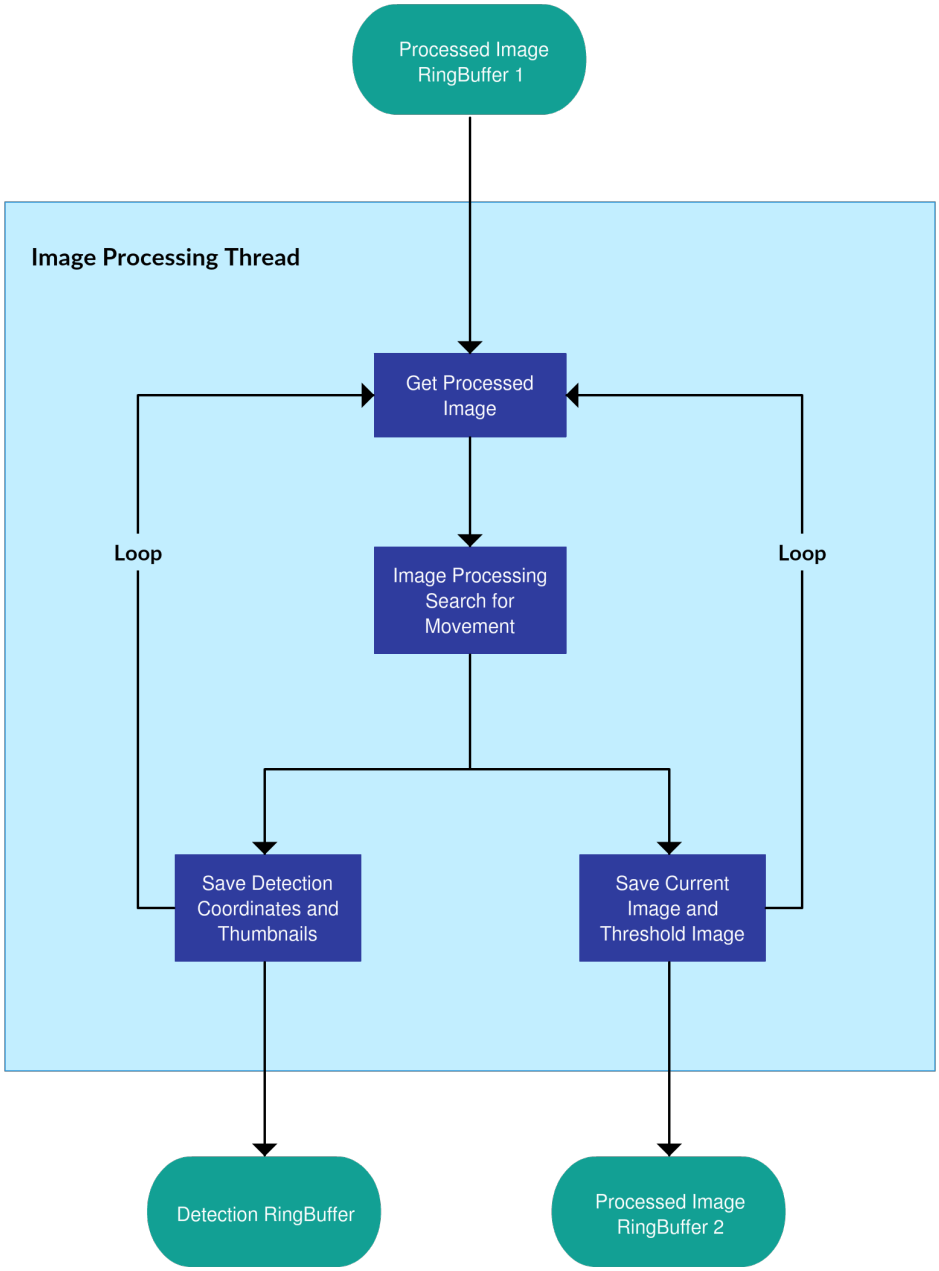


Abbildung 5.5: Softwareablaufdiagramm des Image Processing Threads

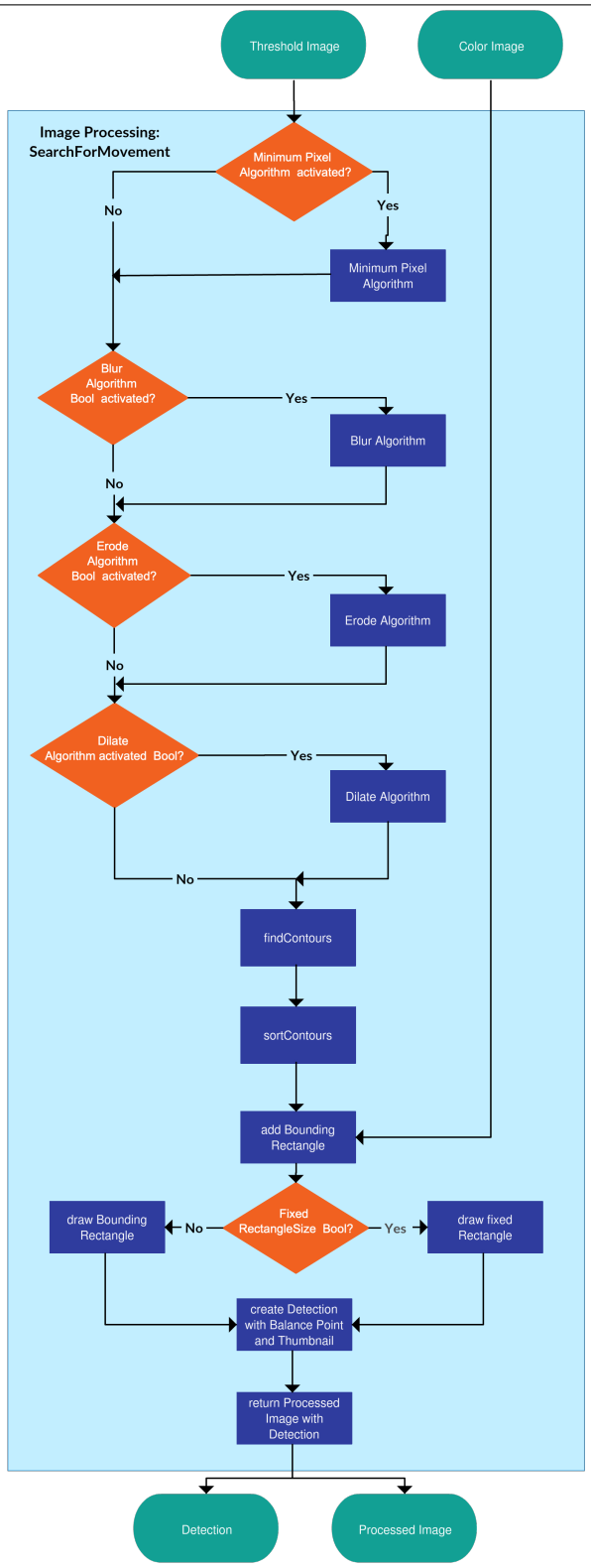


Abbildung 5.6: Softwareablaufdiagramm der SearchForMovement-Methode

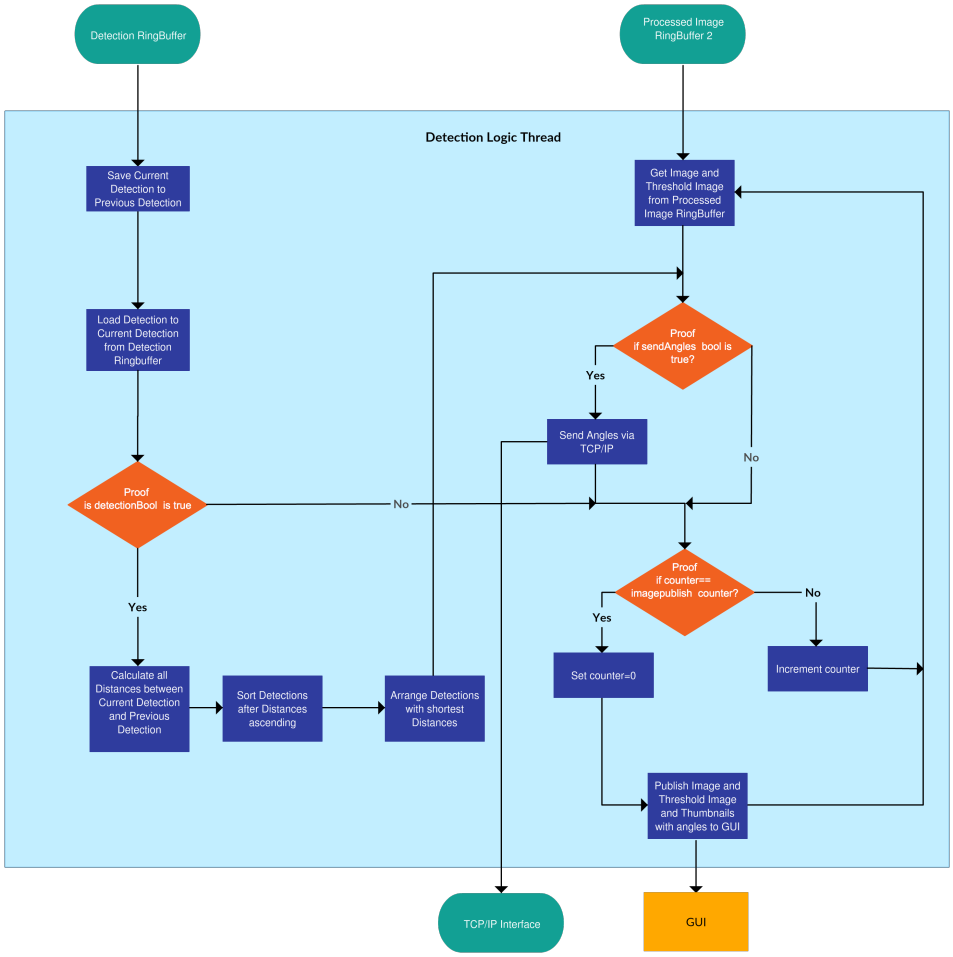


Abbildung 5.7: Softwareablaufdiagramm des Detection Logic Threads

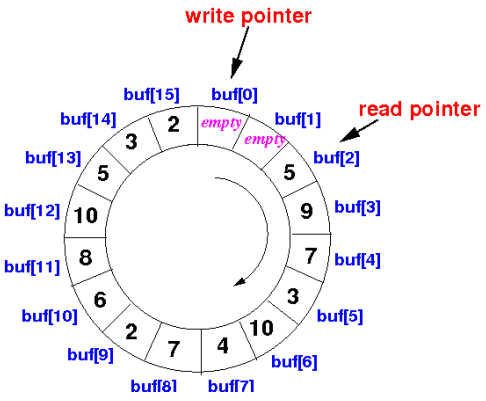


Abbildung 5.8: Aufbau und Funktionsweise eines RingBuffers[rin18]

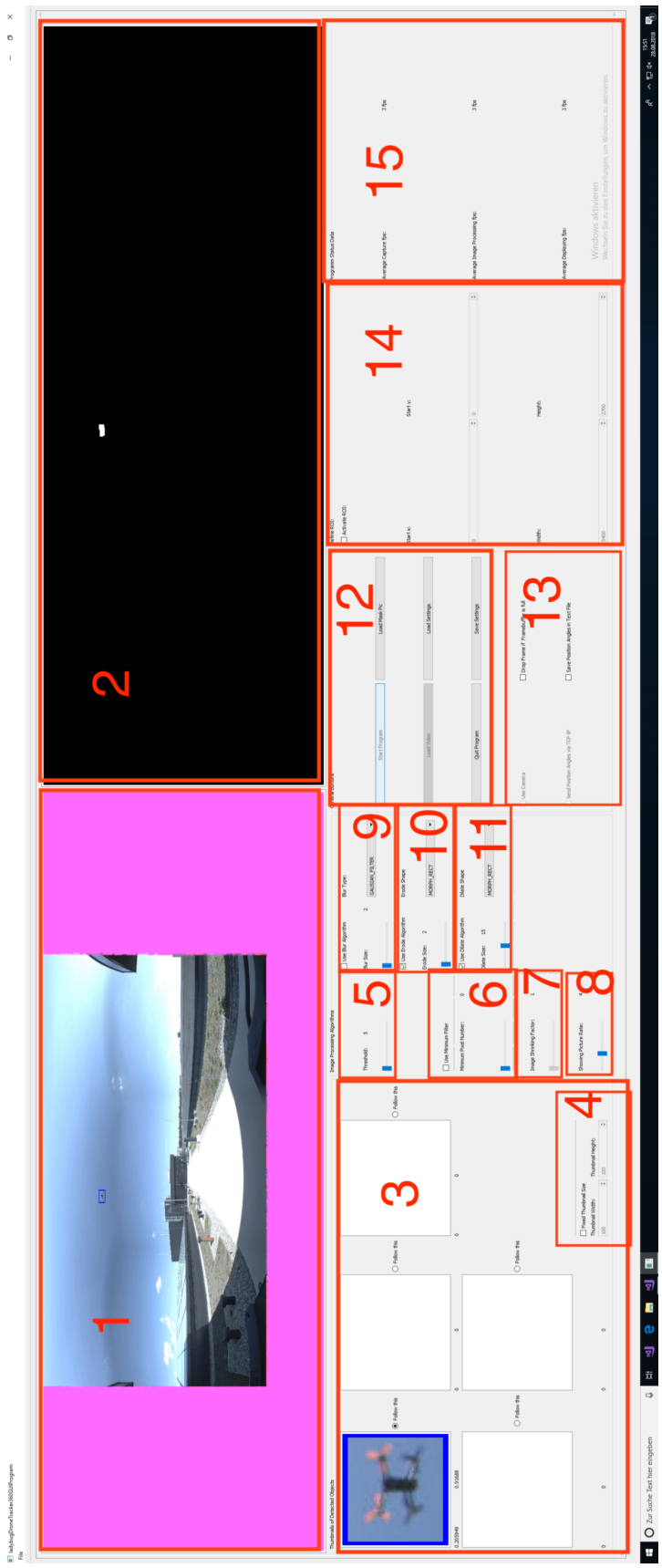


Abbildung 5.9: Übersicht der Grafischen Oberfläche des Programms

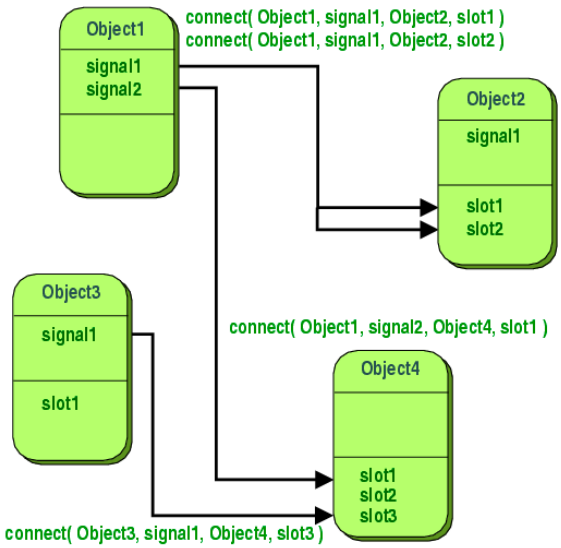


Abbildung 5.10: Signal Slot Übersichtsdiagramm[qsi18]

6 Funktionale Tests

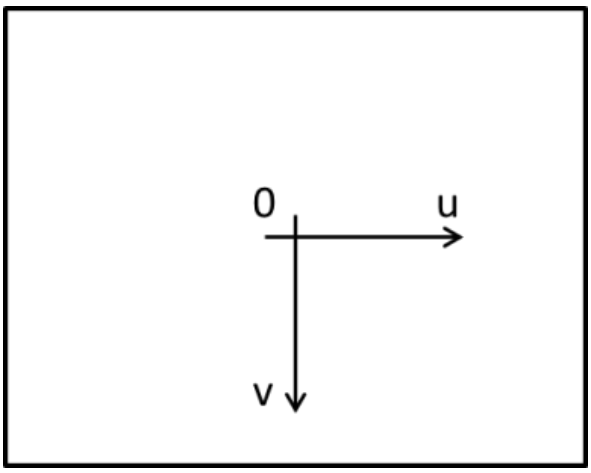
Um die Leistungsfähigkeit des Systems validieren zu können und eine Bewertung für das Erreichen der Anforderungen aus Kapitel 2 machen zu können, müssen mehrere funktionale Tests der Software durchgeführt werden. Im Folgenden wird zuerst die Genauigkeit der Winkelausgabe des Programms geprüft und bewertet. Anschließend wird eine Distanzmessung zur Bestimmung der Erkennungsfähigkeit des Programms durchgeführt und bewertet. Abschließend werden mehrere Drohnentestflüge ausgeführt und die Funktionsfähigkeit des Programms evaluiert.

6.1 Validierung der Winkelausgabe

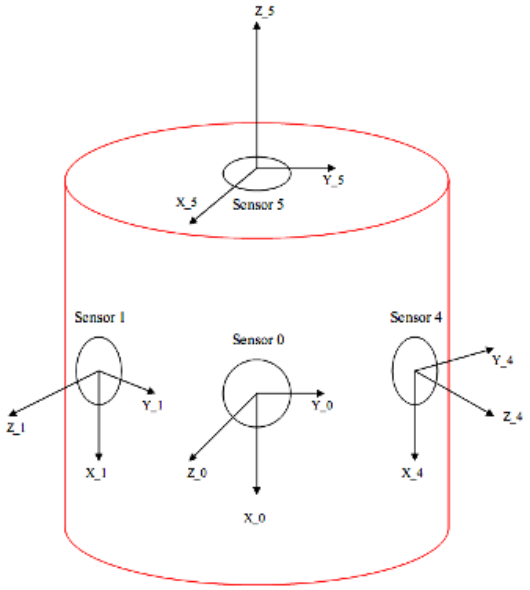
In diesem Test, soll die Ausgabe und Genauigkeit der Winkelberechnung aus Kapitel 4.6 getestet werden, indem die berechneten Winkel, also Azimut und Elevation, mit den berechneten Winkel aus dem vom SDK bereitgestellten Programm ladybugTranslate2dTo3d verglichen werden. Dafür soll zuerst die Genauigkeit des ladybugTranslate2dTo3d-Programms getestet werden. Weiterhin soll durch diesen Test herausgefunden werden, ob jedem Pixel ein spezifischer Winkel zugeordnet werden kann, da es zu Verschiebungen aufgrund des Stitchings kommen kann. Hier soll die Genauigkeit im allgemeinen Bildbereich der Einzelkameras aber auch in den Grenzbereichen, in denen die Bilder zusammengestitcht werden, getestet werden. Das Kamera-SDK wirbt damit, dass eine sehr genaue Positionsbestimmung im Kamerakoordinatensystem möglich ist. Dies erfolgt durch Transformierung der Pixelkoordinaten P(u/v) der Einzelbilder (vergl.6.1 a)) in das jeweilige Kamerakoordinatensystem $P(x_{cam}/y_{cam}/z_{cam})$ (vergl.6.1 b)) . Anschließend wird der Punkt des Kamerakoordinatensystems in das Pixel im Einzelkamera-Koordinatensystem transformiert und danach in das Gesamtkamerakoordinatensystem $P(x_{\_ladybug}/y_{\_ladybug}/z_{\_ladybug})$ (vergl.6.1 c)). Eine genauere Beschreibung hierzu ist in [fli18] zu finden. Die Achsenausrichtung der Einzelkamera-Koordinatensysteme und des Gesamtkamerakoordinatensystems ist in Abbildung 6.1 zu sehen. Ziel ist es eine möglichst genaue Winkelausgabe der detektierten Objekte zu erreichen, die im Sichtfeld der nachführende Schwenk- und Neigeplattform sind, sodass dieses System sich auf das Objekt ausrichten kann.

6.1.1 Aufbau

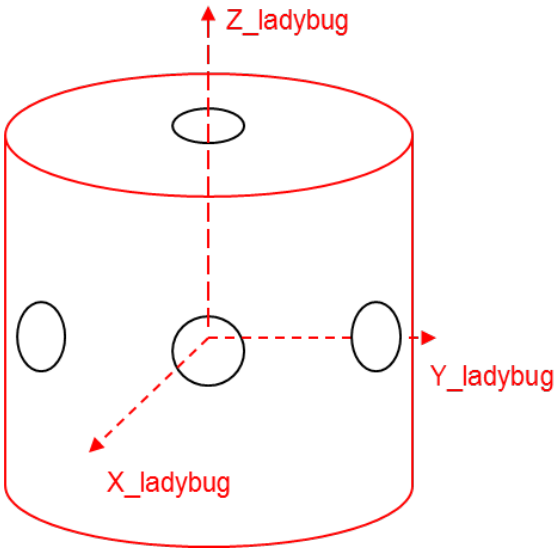
Für den Test wurden als erstes sechs checkerboard registration targets (vergl. [che18]) mit einer Länge und Breite von 17,2 cm ausgedruckt (vergl. Abbildung 6.2. Diese vereinfachen die exakte, pixelgenaue Bestimmung des Mittelpunktes der Targets.



(a) Pixelkoordinaten auf einem Bild der Kamera



(b) Ausrichtung der Koordinatensysteme der Einzelkameras



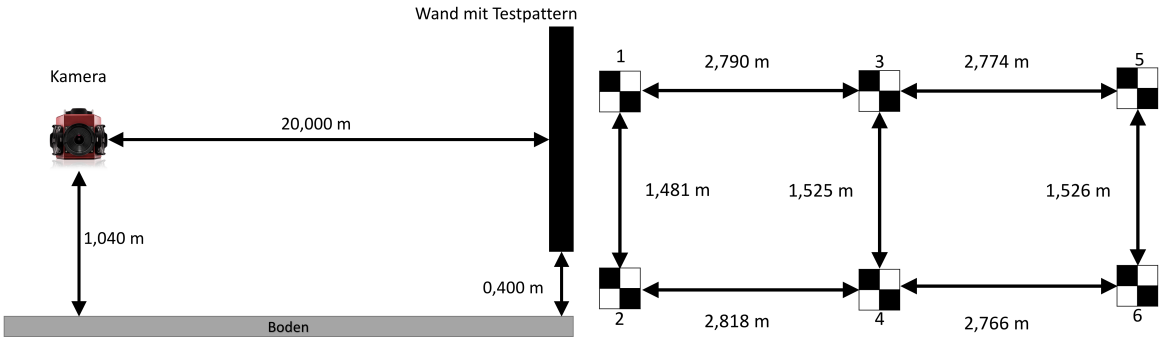
(c) Ausrichtung des Gesamtkamerakoordinatensystems

Abbildung 6.1: Abbildung der einzelnen Koordinatensysteme

Als nächstes wurde die Kamera in der Mitte der Testpattern auf eine Entfernung von 20,00 m zu den Testpattern mit der Nummer 3 und 4 aufgestellt (vergl. 6.3 b)). Hierbei stand die Kamera auf einem Tisch der Höhe 1,04 m. Anschließend wurde mit einem Laserentfernungsmesser vom Typ Bosch GLM 250VF, der Entfernungen mit einer Genauigkeit von ±1 mm misst,der Abstand der Mittelpunkte zwischen

Abbildung 6.2: Versuchsaufbau des Winkelmessungsversuchs auf der Freistrahlstrecke

den einzelnen Testpattern gemessen. Eine Skizze mit den Abständen zwischen den Mittelpunkten der Testpattern sowie der Entfernung der Kamera ist in Abbildung 6.3 dargestellt. Ein Bild des Versuchsaufbaus ist in Abbildung 6.2 dargestellt. Die Abstände wurden willkürlich gewählt. Es wurde lediglich drauf geachtet, dass die Azimut- und Elevationswinkelunterschiede zwischen den einzelnen Aufnahmen in etwa gleich sind. Die Durchführung des Versuchs ist im Folgenden beschrieben.



(a) Abstände des Versuchsaufbau (b) Abstände der Testpattern

Abbildung 6.3: Übersichtsskizze des Winkelmessungsaufbaus

6.1.2 Durchführung

Das Programm LadybugFlyCapturePro wurde geöffnet und die Stitchingentfernung der Kamera auf 20 Meter gestellt, da so Stitchingfehler minimiert werden sollen. Auch wurde die Auflösung der gespeicherten Bilder auf HQLinear gestellt, um eine bestmögliche Auflösung der Bilder von 5400 x 2700 Pixeln ohne Bayer Pattern zu erhalten. Anschließend wurden fünf Messungen gemacht, bei der die Kamera jeweils um einen Winkel von ca. 18 ° gegen den Uhrzeigersinn weitergedreht wurde. Hierbei wurden jeweils die Einzelbilder der Kamera 0 und 1 sowie ein Panoramabild des Gesamtsystems aufgenommen. Anschließend wurde die Kamera 1 in Richtung der Mitte des Testpatterns ausgerichtet und erneut ein Panoramabild und die Einzelbilder der sechs Kameras aufgenommen. Im Anschluss wurde mit Hilfe von Microsoft Paint die Koordinaten der Mittelpunkte der Testpattern im Bild bestimmt. Azimut und Elevation der Testpattern in den Panoramabildern wurden mit der Formeln 4.15 und 4.16 bestimmt. Für die Bestimmung der Winkelpositionen im Gesamtkamerasystem aus den einzelnen Bildern wurde das bereitgestellte Programm von FLIR ladybugTranslate2dTo3d verwendet. Dies transformiert die Pixelkoordinaten der Einzelbilder in die 3D Koordinaten des Kamerakoordinatensystems P($x_{\_ladybug}$/$y_{\_ladybug}$/$z_{\_ladybug}$). Abschließend wurden Azimut λ und Elevation ϕ des Systems mittels folgenden Formeln berechnet:

$$\lambda = atan2(\frac{y_{lady}}{x_{lady}}) \tag{6.1}$$

$$\phi = \arcsin(\frac{z}{\sqrt{x_{lady}^2 + y_{lady}^2 + z_{lady}^2}}) \tag{6.2}$$

Abschließend wurden alle Ergebnisse in eine Microsoft-Excel Tabelle eingetragen und die relativen Abweichungen zwischen den zwei Methoden bestimmt. Hierbei wurde der Winkelfehler durch Differenzbildung bestimmt. Waren Winkelpositionen von Kamera 0 und Kamera 1 berechenbar, so wurde der Mittelwert der Differenzbildung als Abweichung berechnet. Die Formeln für die Messfehlerbildung für den Azimut λ sind in Formel 6.3 und 6.4 zu sehen. Für die Elevation ϕ wurde äquivalent vorgegangen. Auch wurde in Abbildung 6.5 Messung Nr. 5 die 3D-Koordinaten der Messung von Kamera 1 verwendet um die Genauigkeit der Ausgabe der Punkte des Programms ladybugTranslate2dTo3d der Einzelbilder zu überprüfen. Hierbei wurden die Abstände zwischen den einzelnen Testpattern mittels Formel 6.5 berechnet und anschließend mit den gemessenen Werten aus Abbildung 6.3 verglichen. Die berechneten Abstände aus den 3D-Koordinaten sind in Abbildung 6.4 zu sehen.

$$\lambda_{Fehler} = |(\lambda_{Panorama} - \lambda_{Kamera0/1})| \tag{6.3}$$

$$\lambda_{Fehler} = \frac{|(\lambda_{Panorama} - \lambda_{Kamera1})| + |(\lambda_{Panorama} - \lambda_{Kamera0})|}{2} \tag{6.4}$$

$$d_{x_{i,j}} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{6.5}$$

Die Messergebnisse sind in Abbildung 6.5 zu sehen. Die beiden rot-markierten Fehler des Azimutwinkels wurden als falsche Messung betrachtet und nicht in das Endergebnis des durchschnittlichen Azimutwinkelfehlers mit einbezogen, da diese stark von den restlichen Werten abweichen.

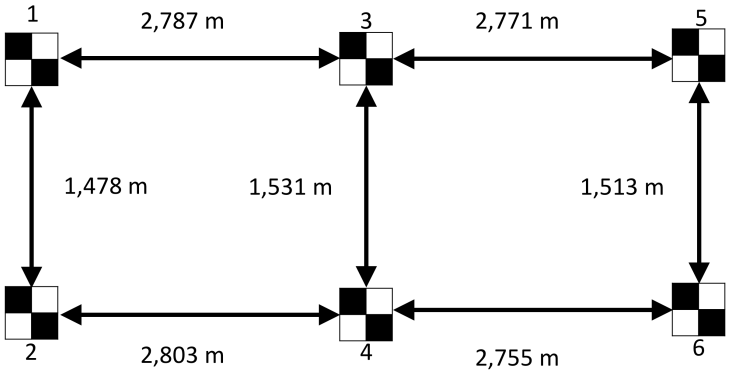


Abbildung 6.4: Berechnete Abstände zwischen den Testpattern

6.1.3 Ergebnis

Aus den einzelnen Kamerabildern wurden die gemessenen Abstände aus Abbildung 6.3 der Testpattern mit den berechneten Abständen aus Abbildung 6.4 verglichen. Hierbei fällt auf, dass die Abstände höchstens um ±2 cm auf einer Entfernung von 20 m abweichen. Daraus lässt sich schließen, dass die 3D-Koordinaten der Einzelbilder sehr genau sind. Somit eignen sich diese sehr gut, um die berechneten Winkel aus den Panoramabild mit den berechneten Winkeln aus den Einzelbildern zu vergleichen.
Beim Vergleich der berechneten Winkel aus den Einzelbildern und dem Panoramabild liegt die durchschnittliche Abweichung zwischen dem berechneten Azimut der Einzelbilder und den berechneten Azimut aus den Panoramabildern bei $\lambda_{Fehler} = 0,0006$ rad und $\phi_{Fehler} = 0,0005$ rad. Auch fällt auf, dass es keine Zunahme der Fehler in Stitchingbereich im Vergleich zu den Einzelbildern gibt, da die berechneten Fehler bei Messung zwei und drei im Vergleich zu den anderen Messungen in etwa gleich bleiben.
Um eine Aussage darüber machen zu können, ob die Abweichungen von der Genauigkeit der Winkelpositionen beim verwendeten Panoramabild ausreichend für das nachführende Kamerasystem sind, wird zu nächst das Sichtfeld der SUN-Kamera berechnet. Das Sichtfeld (Field-of-View FOV) der nachgeführten Kamera, die eine Pixelgröße von p=5,86 µm bei einer Pixelanzahl von n=400 Pixel horizontal und vertikal und einer Brennweite f=25 mm hat, lässt sich wie folgt berechnen:

$$FOV = 2\arctan(\frac{p * n}{2 * f}) = 0,0937rad \tag{6.6}$$

Da die Winkelfehler $\lambda_{Fehler} = 0,0006$ rad und $\phi_{Fehler} = 0,0005$ rad für das System deutlich kleiner als das Sichtfeld des SUN-Systems sind, kann davon ausgegangen werden, dass die detektierten Objekte definitiv im Sichtfeld des ausgerichteten Nachfolgesystems sind. Dies lässt sich jedoch nur für stehende Objekte bestätigen. Bei der Berechnung für bewegende Objekte spielen weitere Faktoren eine Rolle, die im Folgenden erläutert werden. Um eine Aussage machen zu können, ob die nachführende Kamera des SUN eine fliegende Drohne aufgrund der ermittelten Winkelangaben des Programms verfolgen könnte, muss zunächst die Winkelgeschwindigkeit einer Beispieldrohne berechnet werden. Hierfür wird angenommen, dass die Drohne sich auf einer Kreisbahn mit Radius r=15 m um die Kamera mit einer Geschwindigkeit von v= 50 km/h fliegt. Die Winkelgeschwindigkeit der Drohne lässt sich wie folgt berechnen:

$$\omega = \frac{v}{r} = 0,926 \frac{rad}{s} \tag{6.7}$$

Als nächstes muss die Berechnungsdauer des Programms von einem Bild von der Aufnahme bis zur Winkelausgabe verwendet werden. Die Berechnungsdauer des Programms für ein Bild beträgt im Durchschnitt über 100 Messungen t_{prog}= 1,85 s bei einem RingBuffer der Länge 3. Somit lässt sich die maximale Winkeländerung der Drohne zwischen Bildaufnahme und Winkelweitergabe wie folgt berechnen:

$$\lambda_{ges} = \omega * t_{prog} + \lambda_{Fehler} = 1,7137\ rad \tag{6.8}$$

Hier lässt sich erkennen, dass die maximale Berechnungsdauer des Programms deutlich schneller werden muss, um die Drohne bei einem Vorbeiflug im Abstand von 15 m mit einer Geschwindigkeit von 50 km/h noch mit dem nachfolgenden System SUN ausrichten zu können. Durch Umstellen der Formel 6.8 nach t_{prog} und einsetzen von λ_{ges}=FOV=0,0937 rad erhält man die maximale Berechnungsdauer des Programms, die nötig ist, um die Drohne ins Sichtfeld des SUN bekommen zu können:

$$t_{prog} = \frac{\lambda_{ges} - \lambda_{Fehler}}{\omega} = 100\ ms \tag{6.9}$$

Somit muss die nötige Berechnungsdauer des Programms noch auf 100 ms verbessert werden, dass eine Verfolgung einer Drohne im Abstand von 15 Meter mit einer Geschwindigkeit von 50 km/h durchführbar ist.

Messungs-nummer	Kamera 0		Target 1						Kamera 1		Target 1						Pano-rama	Target 1			Fehler Target 1	
	u	v	xlady	ylady	zlady	Azimuth	Elevation		u	v	xlady	ylady	zlady	Azimuth	Elevation		x	y	Azimuth	Elevation	Azimuth	Elevation
Messung 0	1145	1199	19,7637	2,9127	0,9548	6,1369	0,0478			0	0	0	0				2574	1309	6,1366	0,0477	0,0003	0,0001
Messung 1	1146	856	19,8059	-2,6142	0,9438	0,1312	0,0472			0	0	0	0				2812	1309	0,1303	0,0477	0,0009	0,0005
Messung 2	1147	508	18,2998	-8,0186	0,9049	0,4130	0,0453			0	0	0	0				3054	1311	0,4119	0,0454	0,0011	0,0001
Messung 3	1144	96	14,6568	-13,5791	0,8868	0,7472	0,0444		1157	1667	14,6676	-13,5679	0,8801	0,7465	0,0440		3341	1312	0,7458	0,0442	0,0010	0,0000
Messung 4	0	0	0,00	0,00	0,00				1158	1311	10,2107	-17,1745	0,8822	1,0344	0,0441		3589	1311	1,0344	0,0454	0,0000	0,0013
Messung 5	0	0	0,00	0,00	0,00				1157	1260	9,4947	-17,5797	0,8984	1,0756	0,0449		3624	1311	1,0751	0,0454	0,0005	0,0004

Messungsnumm	Kamera 0		Target 2						Kamera 1		Target 2						Panoram	Target 2			Fehler Target 2	
	u	v	xlady	ylady	zlady	Azimuth	Elevation		u	v	xlady	ylady	zlady	Azimuth	Elevation		x	y	Azimuth	Elevation	Azimuth	Elevation
Messung 0	1237	1199	19,7799	2,9118	-0,5269	6,1370	-0,0263			0	0	0	0				2574	1372	6,1366	-0,0256	0,0004	0,0007
Messung 1	1237	855	19,8192	-2,6321	-0,5225	0,1320	-0,0261			0	0	0	0				2813	1372	0,1315	-0,0256	0,0006	0,0005
Messung 2	1240	508	18,3126	-8,0211	-0,5565	0,4128	-0,0278			0	0	0	0				3054	1374	0,4119	-0,0279	0,0009	0,0001
Messung 3	1243	96	14,6694	-13,5824	-0,5709	0,7469	-0,0285		1252	1667	14,6758	-13,5749	-0,5863	0,7464	-0,0293		3342	1375	0,7470	-0,0291	0,0003	0,0002
Messung 4	0	0	0	0	0				1249	1311	10,2190	-17,1827	-0,5723	1,0343	-0,0286		3589	1375	1,0344	-0,0291	0,0001	0,0005
Messung 5	0	0	0	0	0				1249	1259	9,4890	-17,5962	-0,5801	1,0762	-0,0290		3624	1374	1,0751	-0,0279	0,0011	0,0011

Messungsnumm	Kamera 0		Target 3						Kamera 1		Target 3						Panoram	Target 3			Fehler Target 3	
	u	v	xlady	ylady	zlady	Azimuth	Elevation		u	v	xlady	ylady	zlady	Azimuth	Elevation		x	y	Azimuth	Elevation	Azimuth	Elevation
Messung 0	1145	1027	19,9764	0,1422	0,9609	6,2761	0,0481			0	0	0	0				2694	1308	0,0070	0,0489	0,0141	0,0008
Messung 1	1145	683	19,2470	-5,3523	0,9517	0,2712	0,0476			0	0	0	0				2932	1309	0,2699	0,0477	0,0013	0,0001
Messung 2	1145	337	17,0150	-10,4716	0,9144	0,5517	0,0457		1153	1908	17,0213	-10,4617	0,9102	0,5511	0,0455		3174	1310	0,5515	0,0465	0,0001	0,0009
Messung 3	0	0	0,00	0	0				1156	1495	12,6420	-15,4711	0,9083	0,8857	0,0454		3461	1311	0,8855	0,0454	0,0002	0,0001
Messung 4	0	0	0	0	0				1157	1138	7,7204	-18,4280	0,8964	1,1741	0,0448		3709	1311	1,1740	0,0454	0,0000	0,0005
Messung 5	0	0	0	0	0				1156	1087	6,9555	-18,7294	0,9106	1,2152	0,0455		3744	1311	1,2147	0,0454	0,0005	0,0002

Messungsnumm	Kamera 0		Target 4						Kamera 1		Target 4						Panoram	Target 4			Fehler Target 4	
	u	v	xlady	ylady	zlady	Azimuth	Elevation		u	v	xlady	ylady	zlady	Azimuth	Elevation		x	y	Azimuth	Elevation	Azimuth	Elevation
Messung 0	1240	1025	19,9915	0,1083	-0,5743	6,2778	-0,0287			0	0	0	0				2695	1374	0,0058	-0,0279	0,0112	0,0008
Messung 1	1241	681	19,2524	-5,3858	-0,5808	0,2728	-0,0290			0	0	0	0				2934	1375	0,2723	-0,0291	0,0005	0,0000
Messung 2	1244	335	17,0100	-10,5020	-0,6072	0,5531	-0,0304		1256	1906	17,0130	-10,4965	-0,6172	0,5528	-0,0309		3175	1376	0,5527	-0,0303	0,0003	0,0004
Messung 3	0	0	0	0	0				1254	1493	12,6252	-15,4984	-0,6356	0,8872	-0,0318		3462	1377	0,8866	-0,0314	0,0006	0,0004
Messung 4	0	0	0	0	0				1252	1136	7,6988	-18,4480	-0,6331	1,1754	-0,0317		3710	1377	1,1752	-0,0314	0,0003	0,0002
Messung 5	0	0	0	0	0				1251	1085	6,9337	-18,7479	-0,6202	1,2166	-0,0310		3745	1377	1,2159	-0,0314	0,0006	0,0004

Messungsnumm	Kamera 0		Target 5						Kamera 1		Target 5						Panoram	Target 5			Fehler Target 5	
	u	v	xlady	ylady	zlady	Azimuth	Elevation		u	v	xlady	ylady	zlady	Azimuth	Elevation		x	y	Azimuth	Elevation	Azimuth	Elevation
Messung 0	1143	885	19,8594	-2,6321	0,9928	0,1318	0,0495			0	0	0	0				2813	1307	0,1315	0,0500	0,0003	0,0005
Messung 1	1142	512	18,3219	-7,9587	0,9839	0,4098	0,0492			0	0	0	0				3051	1307	0,4084	0,0500	0,0014	0,0008
Messung 2	1141	167	15,4115	-12,7119	0,9455	0,6897	0,0473		1152	1737	15,4113	-12,7118	0,9493	0,6897	0,0475		3292	1307	0,6888	0,0500	0,0009	0,0026
Messung 3	0	0	0	0	0				1155	1323	10,3757	-17,0728	0,9300	1,0247	0,0465		3580	1310	1,0239	0,0465	0,0008	0,0000
Messung 4	0	0	0	0	0				1155	966	5,0966	-19,3178	0,9120	1,3128	0,0456		3828	1310	1,3125	0,0465	0,0004	0,0009
Messung 5	0	0	0	0	0				1154	915	4,2973	-19,5106	0,9321	1,3540	0,0466		3863	1310	1,3532	0,0465	0,0008	0,0001

Messungsnumm	Kamera 0		Target 6						Kamera 1		Target 6						Panoram	Target 6			Fehler Target 6	
	u	v	xlady	ylady	zlady	Azimuth	Elevation		u	v	xlady	ylady	zlady	Azimuth	Elevation		x	y	Azimuth	Elevation	Azimuth	Elevation
Messung 0	1238	855	19,8187	-2,6321	-0,5386	0,1320	-0,0269			0	0	0	0				2814	1372	0,1326	-0,0256	0,0006	0,0003
Messung 1	1239	511	18,3325	-7,9767	-0,5410	0,4104	-0,0271			0	0	0	0				3052	1373	0,4096	-0,0268	0,0008	0,0012
Messung 2	1243	165	15,4059	-12,4707	-0,5775	0,6805	-0,0291		1252	1736	15,4110	-12,7343	-0,5785	0,6906	-0,0289		3293	1374	0,6900	-0,0279	0,0045	0,0011
Messung 3	0	0	0	0	0				1250	1322	10,3710	-17,0908	-0,5874	1,0254	-0,0294		3581	1375	1,0251	-0,0291	0,0003	0,0007
Messung 4	0	0	0	0	0				1249	965	5,0897	-19,3324	-0,5947	1,3134	-0,0297		3829	1375	1,3137	-0,0291	0,0003	0,0000
Messung 5	0	0	0	0	0				1248	914	4,2905	-19,5257	-0,5810	1,3545	-0,0291		3864	1375	1,3544	-0,0291	0,0001	0,0000
																				Durchschnittlicher Fehler:	0,0006	0,0005

Abbildung 6.5: Messdatenübersicht der Winkelmessung

6.2 Validierung der Distanz des Systems

Dieser Test dient zur Validierung der Detektionsdistanz des Systems. In den Hauptzielen von Kapitel 2.1.1 soll eine Detektionsdistanz des Systems von mindestens 30 Metern erreicht werden. Um diesen Test reproduzierbar zu machen, hat man sich dazu entschieden, den Test nicht mit einer fliegenden Drohne durchzuführen, da hier die genaue Distanzbestimmung und Reproduzierbarkeit nicht gegeben ist. Stattdessen wurde im Test eine Drehscheibe mit einem Durchmesser von 10 cm, die halbseitig schwarz und halbseitig weiß ist, verwendet. Zuerst erfolgte eine theoretische Bestimmung der Distanz des Systems aufgrund der Kameraparameter wie Brennweite und Pixelgröße. Anschließend werden der Versuchsaufbau, die Durchführung und das Ergebnis beschrieben.

6.2.1 Theoretische Bestimmung der Reichweite des Systems

Durch die Definition der Drehscheibe als zu erkennendes Objekt lässt sich eine theoretische Reichweite des Systems berechnen. Für eine sichere Detektion einer Bewegung muss sich mindestens ein Pixel zum vorherigen Bild verändern. Da ein Pixel jedoch nicht exakt auf die sich verändernde Fläche eines Kreises ausgerichtet werden kann, wird angenommen, dass mindestens vier Pixel im Halbkreis der Drehscheibe liegen müssen (vergl. Abbildung 6.6), da so sichergestellt ist, dass mindestens ein Pixel sich unabhängig von der Lage des Kreises bezüglich der Pixel verändert. Gegeben sind die Brennweite f, die Pixelgröße p und der Drehscheibenradius r. Die maximale Kantenlänge a eines Quadrats innerhalb eines Halbkreises mit Radius r wird wie folgt bestimmt:

$$geg: \quad p = 3.45\,\mu\text{m};\ f = 4.4\,\text{mm};\ r = 5\,\text{cm}$$

Durch Verwendung des Satz des Pythagoras lässt sich die maximale Seitenlänge eines Quadrats bestimmen:

$$r^2 = a^2 + (\frac{a}{2})^2 \tag{6.10}$$

Durch Zusammenfassung und Umformung nach a, lautet die Formel für die maximale Seitenlänge eines Quadrats in einem Halbkreis:

$$a = \sqrt{\frac{2}{3}} * r = 4.082\,\text{cm} \tag{6.11}$$

$$\frac{a}{2} = 2.041\,\text{cm} \tag{6.12}$$

Da jedoch zwei Pixel übereinander sowie nebeneinander benötigt werden, wird die Seitenlänge auf a/2 halbiert. Nachdem die Seitenlänge der Pixelgröße auf der Distanz berechnet wurde, lässt sich jetzt, mit Hilfe des Strahlensatzes aus Abbildung 6.7 und Formel 6.13, die Distanz des Systems bestimmen. Somit ist eine theoretische Detektion bis zu einer Weite von 26,03 m möglich.

$$d = \frac{f * a}{p} = 26.03\,\text{m} \tag{6.13}$$

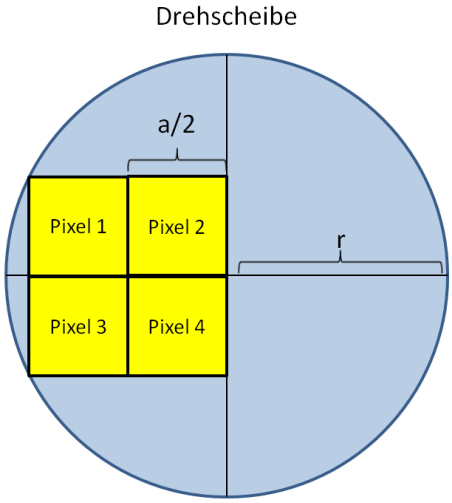


Abbildung 6.6: Skizze zur Berechnung der maximalen Breite eines Quadrats auf der Scheibe

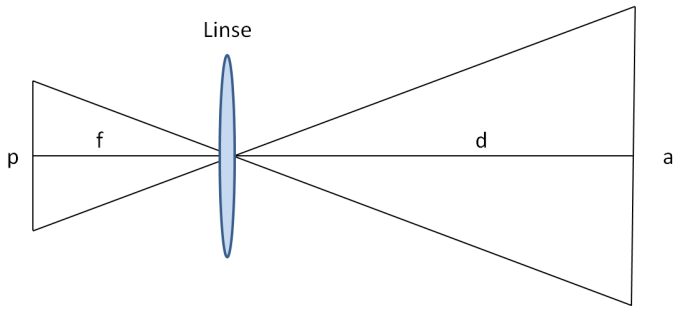


Abbildung 6.7: Skizze zur Berechnung der maximaler Detektionsweite der Kamera

6.2.2 Aufbau

Um die Detektionsdistanz des Systems zu Testen, wurde die oben genannte kreisförmige Drehscheibe mit Durchmesser 10 cm benutzt, die halbseitig schwarz und

halbseitig weiß ist, um einen möglichst großen Kontrast zu erreichen. Diese Drehscheibe wurde auf einen Elektromotor Pololu Stepper Motor NEMA 17 gesteckt, der mit einer Spannung von 10,8 Volt versorgt wurde. Der Drehmotor wurde mit der zugehörigen Steuerung und Software auf einen Laptop betrieben. Hierbei wurde eine Drehfrequenz von 2 Hz eingestellt. Um die Distanz verändern zu können, wurde die Drehschreibe samt Spannungsversorgung auf einen Werkzeugwagen gestellt. Zusätzlich wurde ein Siemensstern (vergl. [sie18b]) mit einem Durchmesser von 28,85 cm und ein US Air Force 1951 Testpattern (vergl. [usa18]) an der Stirnseite des Wagens befestigt, um eine Aussage über die Auflösung des Systems machen zu können. Für die Distanzmessung wurde ein Laserentfernungsmesser vom Typ Bosch GLM 250VF benutzt, der eine Messgenauigkeit von +/- 1 mm und eine Reichweite von 0,05 bis 250 m bereitstellt. Um hier möglichst genau Entfernungen zu messen, wurde ein DIN A4 Blatt Papier neben der Kamera auf einen Stativ angebracht, um die Reflektion der Laserdistanzmessung zu verbessern. Die Kamera wurde auf einem festen Tisch am Ende der Freistrahlstrecke befestigt, um Falschdetektionen durch Erschütterung der Kamera zu verhindern. Die Freistrahlstrecke des DLR Stuttgarts ist ca. 37 Meter lang. Diese wurde als Flugort ausgewählt. Weiterhin wurde der Computer, auf dem die Software betrieben wird, auf einen zusätzlichen Tisch gestellt, um ebenfalls Falschdetektionen durch Erschütterungen des Tisches zu vermeiden. Die Kamera wurde im HQLinear-Modus betrieben und auf der GPU ausgewertet, um eine möglichst hohe Auflösung nutzen zu können. Für die Bildverarbeitungseinstellungen der Software wurde ein Threshold von 5, eine Erosion der Einstellung 1 und eine Dilation der Einstellung 13 verwendet. Weiterhin wurde ein Maskenbild erstellt, das alle Bereiche des Bildes außer die Mitte schwärzt, um so Falschdetektionen durch Bewegungen von ungewünschten Objekten zu vermeiden. Der Versuchsaufbau ist in Abbildung 6.8 zu sehen.

6.2.3 Durchführung

Als erstes wurde eine Aufnahme des Siemenssterns und des US Air Force 1951 Testpatterns in einem Abstand von 1,87 m gemacht. Der Wagen wurde anschließend, beginnend bei vier Meter, um jeweils einen Meter nach hinten verschoben. Hier wurde die Distanz mittels des Laserentfernungsmessers bestimmt. Anschließend wurde bei der jeweiligen Distanz die Anzahl der Detektionen bei 20 aufeinander folgenden Bildern gezählt. Wurde auf mehr als 70% der Bilder die Drehscheibe erkannt, also auf mindestens 14 von 20 Bildern, so wurde das Objekt als detektiert angesehen und der Werkzeugwagen einen Meter weiter von der Kamera wegbewegt. Diese 70% Erkennungsrate wurden gewählt, da es aufgrund der Frequenzüberlagerung von Kamera-Framerate und der Drehfrequenz der Drehscheibe zu Überlagerungen kommen kann, auf denen keine Detektion zu erkennen ist. Wurde die Drehscheibe erkannt, so wurde der Wagen einen Meter nach hintengefahren und erneut die Detektionsanzahl bestimmt.

Abbildung 6.8: Versuchsaufbau des Distanzmessungsversuchs auf der Freistrahlstrecke

6.2.4 Ergebnis

Die Kamera hat im Betrieb mit der Software eine maximale Detektionsweite von 33 m erreicht. Dieser Wert befindet sich somit 6,97 m über der theoretisch bestimmten Detektionsweite von 26,03 m. Dies ist darauf zurückzuführen, dass die theoretische Bestimmung in Kapitel 6.2.1 für ein Worst-Case-Szenario berechnet ist, in dem mindestens vier Pixel auf der jeweiligen Hälfte der Drehscheibe liegen müssen. Dies ist jedoch abhängig von der genauen Positionierung der Kamera. Laut den Hauptzielen in Kapitel 2.1.1 soll eine Minimaldetektionsweite von 30 m erreicht werden. Diese wird durch obigen Test bestätigt. Somit ist die Kamera für eine Detektion von Drohen für eine Reichweite von bis zu 33 m ausgelegt.
Als nächstes wurde zur Bestimmung der Auflösung der Kamera im HQLinear-Modus der Siemensstern betrachtet. Der Siemensstern hat einen Durchmesser von D=28,85 cm und besteht insgesamt aus n=32 hellen und dunkelen Speichen. Der Durchmesser

des Kreises, der nicht aufgelöst werden kann, beträgt auf dem Bild etwa 1,36 cm. Daraus lässt dich die Auflösung der Kamera mit Formel 6.14 berechnet:

$$l = \frac{\pi * d}{n} = 1.50\,\text{mm} \quad [\text{sie18b, sie18a}] \tag{6.14}$$

Somit ist die Auflösung der Kamera etwa 1,50 mm bei einen Abstand von 1,87 m. Auf einen Abstand von 1,00 m hat das System dann eine Auflösung von 0,8 mm, was einer Winkelauflösung von 0,00079 rad entspricht.
Danach wurde die Auflösung mittels des US Air Force 1951 Testpattern bestimmt. Hierzu wird geschaut, welche Linienpaare das System noch auflösen kann. In diesen Fall konnte die Kamera noch Gruppe 0 Element 3 auflösen. Da das Testpattern auf ein DIN A3 Blatt gedruckt wurde und somit die Skalierung verändert wurde, wurden die Linienpaare per Hand mittels Messschieber des Typs Allright 0-150mm/0.01mm 6-Zoll-Vernier mit einer Genauigkeit von 0,01 mm gemessen. Die Auflösung des noch zu erkennenden Linienpaares beträgt demnach 3,90 mm. Die Pixelauflösung beträgt somit 1,95 mm auf einer Distanz von 1,87 m. Auf einem Abstand von 1,00 m hat die Kamera dann eine Auflösung von 1,0 mm, was einer Winkelauflösung von 0,0010 rad entspricht.
Vergleicht man die Auflösung des Siemens-Stern mit der Auflösung des US Air Force 1953 Testpattern so ist eine Abweichung von ca. 0,45 mm zu erkennen. Diese Abweichung resultiert einerseits daraus, dass die vertikalen und horizontalen Linien nicht perfekt auf den Pixeln der Kamera liegen können, womit der Übergang zwischen schwarz und weiß verschwimmen kann, andererseits ist die Ablesung des Innenradius des Siemens-Stern nicht genau. Somit sind die beiden Werte auf Grund ihrer Ungenauigkeit vereinbar. Auch lässt sich durch die bestimmte Auflösung des Systems durch den Siemensstern und das US Air Force 1951 Testpattern das Auflösungsvermögen bei 33 Meter bestimmen. Bei 33 Meter liegt die Pixelauflösung bei 3,3 cm. Dies entspricht in etwa der Größe der Seitenlänge des Quadrates aus Formel 6.12, das das Detektionsprogramm bei einer Auflösung von 33 Metern noch erkennt.

6.3 Flugtest mit einer Drohne

Um eine Aussage über die Funktionsfähigkeit der Software zu machen, wurde ein Drohnenflug mit Hilfe einer Parrot Bebop 2 Drohne durchgeführt. Der Versuchsaufbau, die Durchführung und das Ergebnis sind im Folgenden beschrieben.

6.3.1 Aufbau und Durchführung

Für den Versuchsaufbau wurde die Kamera auf einen Tisch der Höhe 1,04 m gestellt und der Computer auf einen seperaten Tisch, damit keine Störungen bei der Durchführung durch Bewegung des Tisches verursacht werden. Das Programm wurde mit der Kameraauflösung HQLinear betrieben, um eine bestmögliche Auflösung ohne Bayer-Pattern zu garantieren. Die Stitchingdistanz wurde auf 20 m eingestellt. Für die Einstellungen des Threshold, der Dilation und Erosion wurden die ermittelten Werte aus Kapitel 4.4 gewählt. Weiterhin wurde ein Maskenbild erstellt, um unnötige Bereiche auszuschließen und Falschdetektionen von sich bewegenden Objekten, wie z.B. sich bewegende Gräser, zu verhindern. Anschließend wurde mit der Drohne Parrot Bebop 2 (vergl. [beb18]) vor der Kamera gestartet und eine Flugroute im Abstand von 0 bis 30 Metern zufällig abgeflogen. Hierbei wurden unterschiedliche Höhen und Abstände abgeflogen. Es wurden vier Testflüge unternommen. Zwei Testflüge haben mit dem ladybugDroneTracker360GUIProgramm stattgefunden und werden im Folgenden als Liveflüge bezeichnet. Zwei weitere wurden mit Hilfe des LadybugCapPro-Programms aufgenommen und werden in folgenden als Offlineflüge bezeichnet. Um die Anzahl der Detektionen zu bestimmen, wurden die Einzelbilder am Ende der Verarbeitungskette des Programms nach Ausführen des DetectionLogic Threads abgespeichert. Schließlich wurden die Einzelbilder ausgewertet, indem die Detektionen der Drohne in den Einzelbildern gezählt wurden. Als Maß des Erfolgs für den Versuch wurde die prozentuale Erkennungsrate der Drohne zu der Gesamtmenge an aufgenommenen Bildern in Betracht gezogen. Bei einer Detektion der Drohne wird ein ROI-Rechteck um die Drohne im Einzelbild gezeichnet. Anschließend wurden, um den Flug reproduzierbar zu machen, die Einzelbilder in ein Video umgewandelt. Somit kann das Video später immer wieder im Video Mode als Test in das Programm geladen werden um z.B. den Flug erneut mit anderen Bildverarbeitungsparametern zu testen. Ein Beispielbild des Testfluges inklusive entdeckter Drohne ist in Abbildung 6.9 zu sehen.

6.3.2 Ergebnis

Um eine Aussage über die Funktionsfähigkeit des Programms machen zu können, wurden die Bilder gezählt, auf denen die Drohne mit einer ROI Markierung gekennzeichnet wurde und das Verhältnis zu der Anzahl der Gesamtbilder der einzelnen Videos oder Liveaufnahmen berechnet. Für die Auswertung der Offlineflüge wurden die Videos über den Videomode des ladybugDroneTracker360GUIProgramm geladen und anschließend genauso ausgewertet wie die Liveflüge. Die Tabelle 6.1 zeigt

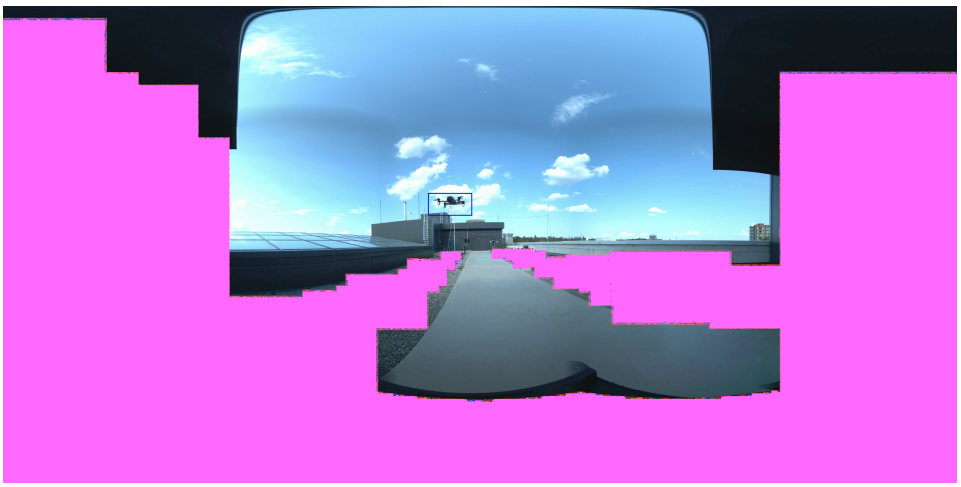


Abbildung 6.9: Beispielbild des Testfluges der Drohne

die einzelnen Testflüge, die Anzahl der Bilder, in denen die Drohne nicht erkannt wurde, die Gesamtanzahl der Bilder jedes Videos und das prozentuale Verhältnis der erkannten Drohnen in den Bildern zur Gesamtanzahl der Bilder jedes Videos.

Flugname	Bilder mit nicht erkannter Drohne	Gesamtbilder	Erkennungsrate
Liveflug 1	15	133	88,72 %
Liveflug 2	15	163	90,80 %
Offlineflug 1	3	205	98,54 %
Offlineflug 2	3	767	99,61 %

Tabelle 6.1: Auswertung der nicht erkannten Detektionen

Beim Auswerten der Flüge fällt auf, dass die Liveflüge im Erkennungsverhältnis deutlich schlechter sind, wie die Offlineflüge. Dies ist auf die höhere Framerate von ca. 10 Frames pro Sekunde bei dem Video des Offlinefluges im Vergleich zu ca. 4 Frames pro Sekunde beim Liveflug zurückzuführen. Weiterhin wurde in allen Flügen jedoch die Grenze von 80% Erkennungsverhältnis überschritten, was einen großen Erfolg darstellt. Bei den Liveflügen traten die Bilder, in denen die Drohne nicht erkannt wurde, meist in den Grenzbereichen ab 30 Meter auf. Auch hier fiel auf, dass die Drohne höchstens 2 bis 3 Frames hintereinander nicht erkannt wurde. Um dies zu verbessern, könnte die letzte Position einer Detection für 4 bis 5 Frames gemerkt werden, falls keine Nachfolgedetection vom Algorithmus aus Kapitel 4.7 zugewiesen wird, wodurch eine unterbrechungsfreie Verfolgung besser möglich wäre. Bei näheren Distanzen unter 30 Meter erkennt das Programm zwischendurch die Drohne auf einzelnen Frames nicht. Dies ist aufgrund der Seltenheit des Auftretens zu vernachlässigen und kann durch oben genannten Lösungsansatz ebenso verhindert werden.

Beim Auswerten der Offlineflüge fällt auf, dass das Programm eine Fehlerrate von unter 1,5% hat. Dies ist ein ausgezeichnetes Ergebnis, welches auf die höhere Framerate in Vergleich zu den Liveflügen zurückzuführen ist. Hier sind die einzelnen Bilder, auf denen die Drohne nicht erkannt wurde, lediglich sehr selten und treten nicht in mehreren Frames in Folge auf. Auch hier können diese fehlenden Detektionen durch das Merken der letzten Position über mehrere Frames zusätzlich verbessert werden. Zusammengefasst lässt sich festhalten, dass die Tests mit den Liveflügen und Offlineflügen ein sehr großer Erfolg waren. Die Drohne wurde bei einer Gesamtzahl von 1268 ausgewerteten Bildern lediglich auf 36 Bildern nicht erkannt, was einem Erkennungsverhältnis von 97,1% entspricht. Somit wird die Drohne in der definierten Detektionsweite mit einer sehr hohen Wahrscheinlichkeit erkannt.

6.4 Flugtest mit mehreren Drohnen

Dieser Test soll die Funktionsfähigkeit des Verfolgungsalgorithmus und das Verhalten des Programms beim Flug mit mehreren Drohnen prüfen. Der Versuchsaufbau, die Durchführung und das Ergebnis sind im Folgenden beschrieben.

6.4.1 Aufbau und Durchführung

Der Versuchsaufbau und die Durchführung sind äquivalent zu Kapitel 6.3.1. In diesem Versuch wurden jedoch einmal zwei und einmal drei Parrot Bebop 1 Drohnen verwendet und mit diesen eine beliebige Route im Bereich 0 bis 30 m abgeflogen. Weiterhin wurden die beiden Videos mit dem LadybugCapPro-Programm aufgenommen und anschließend offline mit dem ladybugDroneTracker360GUIProgramm mit der Auflösung HQLinear und der Stitchingdistanz 20 m analysiert. Es wurden sowohl die Anzahl der nicht erkannten Drohnen in den Einzelframes gezählt, als auch die Anzahl der Sprünge der Drohnen in dem Thumbnail-Fenster aus Kapitel 5.4.1. Ein Sprung einer Drohne im Thumbnail-Fenster bedeutet, dass der Verfolgungsalgorithmus aus Kapitel 4.7 in zwei aufeinanderfolgenden Detektionen die ROIs falsch zugeordnet hat.

6.4.2 Ergebnis

Nachdem die Frames gezählt wurden, in denen eine Drohne nicht erkannt wurde, wurden diese zur Analyse in Tabelle 6.2 eingetragen. Bei einem Flug mit n Drohnen und n_{frames} aufgenommenen Frames berechnet sich die Anzahl an möglichen Drohnendetektionen $n_{detections}$ mit Formel 6.15. Die Anzahl der Ist-Detektionen wurde im Verhältnis zur Anzahl der möglichen Drohnendetektionen gestellt.

$$n_{detections} = n * n_{frames} \tag{6.15}$$

Sowohl beim Flug mit zwei Drohnen als auch mit drei Drohnen besteht eine Detektionsrate von über 90 %, beim Flug mit zwei Drohnen sogar 100 %. Die niedrigere

Flugname	Frames	Ist-Detektionen	Maximal mögliche Detektionen	Erkennungsrate
Flug mit 2 Drohnen	223	446	446	100,00 %
Flug mit 3 Drohnen	351	987	1053	93,73 %

Tabelle 6.2: Auswertung der möglichen Detektionen

Detektionsrate beim Flug mit drei Drohnen ist darauf zurückzuführen, dass eines der drei Drohnen sich zwischenzeitlich außerhalb der Detektionsreichweite befunden hat. Hierdurch wurden 46 der nicht erkannten Detektionen ausgelöst, sonst hätte der Flug mit drei Drohnen eine deutlich verbesserte Detektionsrate von 98,10 %. Weiterhin konnte nachgewiesen werden, dass praktisch kein Zusammenhang der Detektionsrate mit der Anzahl der fliegenden Drohnen besteht, solange diese geringer als fünf ist, da dies die maximale Anzahl an zu erkennenden Objekten ist.

Als nächstes wurde die Anzahl der Sprünge der Drohnen im Thumbnails-Feld der grafischen Oberfläche gezählt. Die Anzahl der Sprünge im Vergleich zur maximalen Anzahl an möglichen Sprüngen ist in Tabelle 6.3 dargestellt. Hierbei errechnet sich die Anzahl der maximalen Sprünge der Thumbnails k mit der Frameanzahl n_{Frame} und der Anzahl der Drohnen n_{Drone} wie in Formel 6.16.

$$k = n_{Frame} * n_{Drone} \tag{6.16}$$

Hier wurde ebenfalls die Anzahl der Sprünge der Thumbnails aus beiden Flügen im Verhältnis zur Anzahl der möglichen Sprünge gestellt. Dies ist in Tabelle 6.3 dargestellt. Ein Beispiel der drei Drohnen in den Thumbnailfenstern des Programms ist in Abbildung dargestellt.

Flugname	Frames	Ist-Sprünge	Maximale Sprünge	Sprungrate
Flug mit 2 Drohnen	223	6	446	1,35 %
Flug mit 3 Drohnen	351	11	1053	1,04 %

Tabelle 6.3: Auswertung der Sprungrate der Thumbnails

Auch hier fällt auf, dass die Sprunganzahl der Thumbnails in den einzelnen Frames sehr gering ist. Beim Flug mit zwei Drohnen sind zwei Sprünge bedingt dadurch, dass sich die Drohnenflugbahnen einmal kreuzen und somit die Drohnen als ein bewegendes Objekt wahrgenommen werden und diese sich anschließend wieder

trennen. Beim Flug mit drei Drohnen ist dies ebenfalls drei Mal der Fall gewesen, d.h. bei insgesamt sechs Sprüngen. Ansonsten ist die Zuordnung der Thumbnailfenster sehr konstant und zuverlässig. Hier fällt auf, dass der Algorithmus vor allem Fehler produziert, wenn zwei Drohnen kreuzende Flugbahnen haben und somit durch die Dilation die beiden Bereiche des Thresholdbildes zusammengefasst werden. Dies war aufgrund der Simplizität des Algorithmus, zu erwarten. Es stellt jedoch kein Problem dar, da die Objektverfolgung lediglich ein Kann-Kriterium in Kapitel 2.2.1 darstellt. Weiterhin funktioniert sie für normale Flugbahnen ohne Kreuzung sehr gut. Die Sprungraten kleiner 2% beweisen dies. Somit sollte eine problemslose Verfolgung eines in der grafischen Oberfläche ausgewählten Objektes bei einfachen Flugbahnen mittels des nachführenden Systems möglich sein.

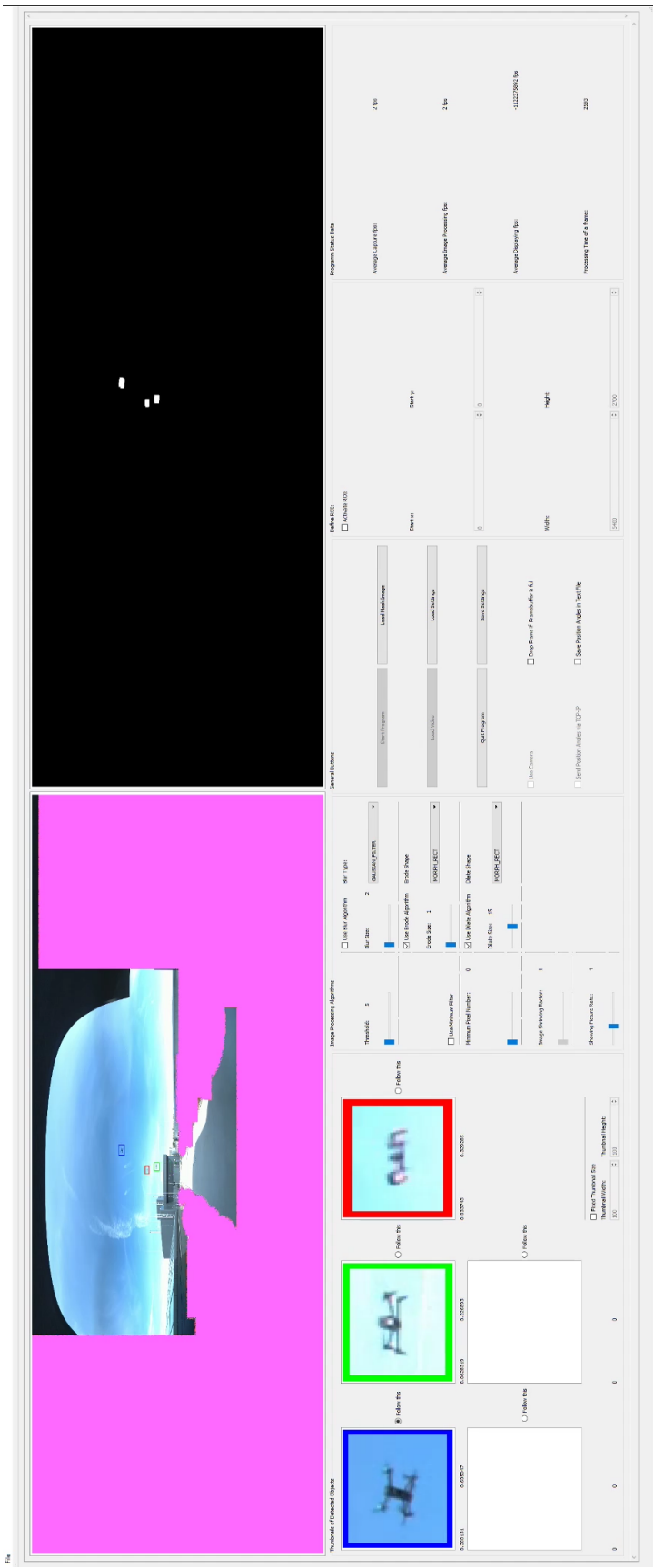


Abbildung 6.10: Beispielbild des Versuchs mit drei Drohnen

7 Fazit und Ausblick

Im folgenden Kapitel, werden die einzelnen Arbeitschritte zusammengefasst und der Erfolg des Projekts bewertet. Abschließend werden Verbesserungsvorschläge aufgezeigt, die im Folgeprojekt verwirklicht werden können.

7.1 Zusammenfassung

Zu Beginn wurden unterschiedliche Bildverarbeitungsalgorithmen getestet. Anschließend wurde auf Basis von Liveflügen in Kapitel 4.4 ein geeignetes Parameterset zum Betrieb des Programms für Drohnendetektion ausgewählt. Weiterhin wurde das Programm objektorientiert aufgebaut, so dass jederzeit die Bildverarbeitungs- und Detektionsverfolgungsalgorithmen durch neue, bessere ersetzt werden können. Somit erleichtert dies die Weiterentwicklung des Programms.
Auch wurde eine benutzerfreundliche und leicht verständliche grafische Oberfläche zur einfachen Interaktion mit dem Benutzer auf Basis des Qt-Frameworks umgesetzt. Somit kann das Programm nach kurzer Einweisung von Personen leicht benutzt werden und die Auswirkungen des Umstellens verschiedener Parameter auf die Bildverarbeitung leicht demonstriert werden. Dies unterstützt die Demonstrationsfähigkeit auf Messen oder an Tagen der offenen Tür für Schulen und sonstige Bildungseinrichtungen. Weiterhin kann so eine leichte Anpassung der Parameter für unterschiedliche Orte und Belichtungsverhältnisse erfolgen.
Auch wurde die Verarbeitungsgeschwindigkeit des Programm anschließend optimiert durch Auslagerung der Bildverarbeitungsalgorithmen auf die Grafikkarte mit Hilfe des CUDA-Frameworks. Weiterhin wurden die Aufgaben des Programms so auf Threads aufgeteilt, dass alle Threads möglichst die gleiche Dauer haben und somit der Ablauf möglichst schnell erfolgt.
Abschließend wurden in Kapitel 6.1 zuerst die Winkelausgabe von Azimut und Elevation und deren Genauigkeit getestet. Hierbei war der Azimuthwinkel und Elevationswinkel durch das Panoramabild ausreichend genau mit einer mittleren Abweichung beim Azimutwinkel von 0,006 rad und beim Elevationswinkel von 0,0005 rad. Auch fiel hier auf, dass die Bearbeitungsdauer der Einzelbilder durch das Programm zu gering ist, um eine Verfolgung einer Drohne mit einer Geschwindigkeit von 50 km/h im Abstand von 15 Metern auf einer Kreisbahn mit dem nachfolgenden SUN-System gewährleisten zu können. Hier muss also im Nachhinein eine Erhöhung der Framerate erfolgen. Diese Verbesserungsvorschläge sind in Kapitel 7.2 aufgeführt.
Anschließend wurde im Kapitel 6.2.1 die Reichweite des Systems getestet. Hier wurde zuerst die theoretische Reichweite des Systems berechnet. Anschließend erfolgte die

praktische Reichweitenbestimmung mittels einer Drehscheibe. Hierbei konnte nachgewiesen werden, dass eine Detektionsreichweite von 33 m möglich ist. Dies übertrifft die Anforderungen des Systems aus Kapitel 2.2.
In den abschließenden Flugtests konnte nachgewiesen werden, dass das System die Drohne sowohl im Liveflug als auch im Offlineflug sehr gut erkennt und verfolgen kann, was ein Erkennungsverhältnis aller ausgewerteten Frames von 97,1 % unterstreicht. Hier zeigte sich jedoch auch, dass gerade das Erkennungsverhältnis deutlich besser ist, wenn das Programm eine höhere Framerate, also eine kürzere Verarbeitungsdauer, aufweisen kann. Weiterhin wurde der Verfolgungsalgorithmus in Kapitel 6.4 getestet. Auch dieser funktionierte sehr gut mit einen Sprungverhältnis geringer als 2 %. Dieser Algorithmus geriet lediglich an seine Grenzen, wenn die Drohnen sehr nahe oder sich kreuzende Flugbahnen hatten. Weiterhin muss abschließend erwähnt werden, dass in den Videos auch Fehldetektionen stattgefunden haben. Es wurden sowohl Flugzeuge, Schatten der Drohnen am Boden und sich bewegende Gräßer, die nicht durch das Maskenbild bedeckt waren, erkannt. Dies ist auf die Wahl der Durchführung des Programm mittles Differenzbildmethode (vergl. Kapitel 4.2) zurückzuführen, da alle Bildveränderungen unabhängig vom Objekt erkannt werden. Vorschläge wie bestehende Mängel des Programms verbessert werden, sind im Folgenden aufgeführt.

7.2 Verbesserungsvorschläge des Programms

Um die Funktionalität des Programms zukünftig zu verbessern, werden im Folgenden einige Verbesserungsvorschläge aufgeführt und erläutert.

Das größte Problem des Programms ist, dass die Framerate des Programms nicht den geforderten 10 Hz entspricht. Dies kann jedoch durch verschiedene Vorschläge verbessert werden.
Der Hauptgrund für die langsame Verarbeitung der Einzelbilder ist die Größe der Bilddateien von 5400x2700 Pixeln im Panoramabild. Da eine Auflösungsverringerung auch die Detektionsweite verringern würde, könnte man das Bild jedoch in z.B. sechs gleichgroße Teilbilder aufteilen und diese einzeln und parallel in unterschiedlichen Threads verarbeiten und am Ende im DetectionLogic-Thread wieder zusammensetzen und analysieren. Dies würde jedoch das Einfügen von einer zusätzlichen Einheit erfordern, die Bereiche richtig zusammenfasst, die in den Grenzbereichen der Bilder liegen. Die theoretische Bearbeitungszeit eines Bearbeitungsschrittes des einzelnen Panoramabildes liegt bei $O(5400x2700) = O(14580000)$ Ausführungsschritten. Die theoretische Bearbeitungszeit eines Bearbeitungsschrittes der sechs Einzelbilder liegt jedoch nur bei $O(5400/3x2700/2) = O(2430000)$. Wenn die Bearbeitung der sechs Einzelbilder parallelisiert werden kann, wäre das ein erheblicher Geschwindigkeitsgewinn.

Ein weiterer Verbesserungsvorschlag ist, auf die Verwendung des Panorama-Bildes zu verzichten, so kann einerseits Zeit zur Berechnung des Stitchings eingespart werden, andererseits erfolgt, wie bereits vorher genannt, die Verarbeitung der sechs Einzelbilder in gesonderten Threads, wodurch die Bildverarbeitung des Programms an Geschwindigkeit gewinnt. Ein weiterer Vorteil dieses Vorschlags ist, dass somit Azimut und Elevation mit den Koordinaten des Kamerakoordinatensystems auf Basis der Rechnung des Beispielprogramms ladybugTranslate2dTo3d erfolgen kann. Dadurch wäre eine erhöhte Genauigkeit der Winkelberechnung garantiert, da alle Pixelkoordinaten zuerst in das kalibrierte Kamerakoordinatensystem umgewandelt werden.

Zur Verbesserung des Detektionsverfolgungsalgorithmuses könnte, zum Vergleich zweier Detektionen und das Erleichtern der Zuordnen der Detektionen, z.B. ein Vergleichsvektor erstellt werden. In diesem Vektor könnten nicht nur die Schwerpunkte der Detektionen miteinander verglichen werden, sondern auch andere Größen, wie z.B. Abstand, Farbe und Größe der ROIs zusätzlich in die Entscheidung mit eingehen.

Ein weiterer Verbesserungsvorschlag für die Objektverfolgung ist das Merken der Koordinaten und deren Zuordnung der verschiedenen erkannten Objekte über mehrere Bildfolgen hinweg. Somit könnte einerseits die Position bei Verlust der Detektion über 3-4 Bildfolgen gemerkt werden, in der Annahme, dass das Objekt in der Nähe wieder auftaucht, andererseits könnte ein Kalman-Filter umgesetzt werden, der die nächste wahrscheinliche Position bestimmt und somit einen weiteren Parameter für die Objektzuordnung bereitstellt. Dies wurde jedoch aufgrund der beschränkten Zeit des Projekts nicht mehr umgesetzt.

Auch empfiehlt sich für das Verhindern von Falschdetektionen, wie z.B. Flugzeugen oder Vögel, ein neuronales Netzwerk zu verwenden, das zwischen Drohen und Nicht-Drohnen unterscheiden kann. Hierfür wurde die Möglichkeit eine gleichbleibende ROI-Fläche auszuwählen in der GUI umgesetzt.

7.3 Ausblick

Nach den Bewertungen des Erreichens der Anforderungen aus Kapitel 2.2 lässt sich erkennen, dass ein Großteil der wichtigsten Anforderungen erfolgreich umgesetzt wurde. Lediglich die Anforderungen 1200 und 1250 aus Kapitel 2.2.1, bei denen es sich um muss-Anforderungen handelt, sind nicht erreicht worden. Diese sollten im Nachhinein noch verbessert werden. Weiterhin wurden durch die funktionalen Test die Eignung des Programms für die Drohnendetektion von bis zu 33 m nachgewiesen. In den Flugtests mit der Drohne Parrot Bebop 2 wurde nachgewiesen, dass das Erkennungsverhältnis der aufgenommenen Bilder im Schnitt bei über 97 % liegt, was den Gesamterfolg des Projekts unterstreicht und beweist. Weiterhin wurde bewiesen, dass das System die Fähigkeit besitzt, mehrere Drohnen zu verfolgen. Dies wurde mit einem Sprungverhältnis der Thumbnails in der GUI geringer als 2,00% unterstrichen.

Somit war das Projekt für den Erstentwurf einer Software für die Drohnendetektion mit Hilfe eines 360°-Kamerasystems, in dem vorgegebenen Zeitraum von sechs Monaten, ein voller Erfolg, auch wenn einige Verbesserungen im Folgeprojekt umgesetzt werden können.

Literaturverzeichnis

[beb18] *Drohne Parrot Bebop 2*. https://www.parrot.com/de/drohnen/parrot-bebop-2, 2018. – Besucht: 15.09.2018

[che18] *Checkerboard Registration Targets*. https://knowledge.faro.com/Hardware/3D_Scanners/Focus/Checkerboard_Registration_Targets_Download_for_the_Laser_Scanner_or_Hand-Held_Scanner, 2018. – Besucht: 15.09.2018

[cud18a] *CUDA Toolkit*. "https://developer.nvidia.com/cuda-toolkit, 2018. – Besucht: 08.08.2018

[cud18b] *CUDA Zone*. "https://developer.nvidia.com/cuda-zone, 2018. – Besucht: 31.07.2018

[dif18] *Bewegungserkennung mit Differenzbildern*. "https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2009-aw2/bernin/folien.pdf, 2018. – Besucht: 03.08.2018

[dji18a] *Best Follow Me Drones (With Video Comparisons)*. https://store.dji.com/guides/camera-drone-that-follows-you/, 2018. – Besucht: 01.10.2018

[dji18b] *HOW TO: Setup and use Waypoints*. https://forum.dji.com/thread-30373-1-1.html, 2018. – Besucht: 01.10.2018

[dji18c] *Phantom 4 Advanced*. https://www.dji.com/de/phantom-4-adv, 2018. – Besucht: 01.10.2018

[dji18d] *Phantom 4 Pro V2.0*. https://store.dji.com/de/product/phantom-4-pro-v2?ch=Text&pbc=fXB2Y5Fk&from=dap_product&as=0081&pm=link&vid=43151, 2018. – Besucht: 01.10.2018

[dji18e] *You Can Film Like a Pro with DJI Drone "ActiveTrack"!* https://store.dji.com/guides/film-like-a-pro-with-activetrack/, 2018. – Besucht: 01.10.2018

[Dre04] Drexler, Sebastian: *Kurzeinführung in TCP/IP*. 21.06.2004. – Universität Erlangen

[dro18] *Verbraucher schicken Drohnen auf Höhenflug*. https://de.statista.com/infografik/5935/entwicklung-des-weltweiten-drohnen-marktes-fuer-endverbraucher/, 2018. – Besucht: 31.07.2018

[ent18] *Bewegungserkennung mit Differenzbildern*. "https://www.kompf.de/gps/distcalc.html, 2018. – Besucht: 16.08.2018

[Erh08] Erhardt, Angelika: *Einführung in die Digitale Bildverarbeitung - Grundlagen, Systeme und Anwendungen*. Berlin Heidelberg New York : Springer-Verlag, 2008. – ISBN 978–3–519–00478–3

[fin18] *findContours*. "https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html, 2018. – Besucht: 27.08.2018

[fli18] *Geometric Vision using Ladybug Cameras*. https://eu.ptgrey.com/KB/10621, 2018. – Besucht: 19.09.2018

[Fol08] Folkers, Christian: *High Performance Realtime Vision for Mobile Robots on the GPU - Eine Implementierung für den Robocup Weingearten -*. 2008

[gre18] *Umweltaktivisten fliegen Drohne gegen AKW*. https://www.zdf.de/nachrichten/heute/greenpeace-aktivisten-superman-drohne-fliegt-gegen-akw-100.html, 2018. – Besucht: 15.08.2018

[hsv18] *Bewegungserkennung mit Differenzbildern*. "https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2009-aw2/bernin/folien.pdf, 2018. – Besucht: 03.08.2018

[IT16] Igor Tchouchenkov, Rainer S.: *Eigenschaften, Detektions- und Abwehrmöglichkeiten von kleinen UAS*. 16.03.2016. – Counter-UAS Kolloquium

[Kar06] Karakoc, Gürcan: *Morphologische Bildoperationen*. 2006

[Kop07] Kopf, Dr. S.: *Objekterkennung durch Vergleich von Farben*. 2007. – HWS2007

[Kri13] Kriehs, Robert: *Entwurf und Implementierung einer Kalibrierungsfreien Computer Vision für die FUmanoids*. 2013

[Kü16] Kühne, Jens: *Automatic Radio-controlled Drone Identification Solution*. 16.03.2016. – Counter-UAS Kolloquium

[lad18a] *FLIR Ladybug 5+ Technical Reference*. https://www.ptgrey.com/support/downloads/10128, 2018. – Besucht: 01.08.2018

[lad18b] *Ladybug 5+ Datasheet*. https://www.ptgrey.com/support/downloads/10752, 2018. – Besucht: 31.07.2018

[lad18c] *Ladybug SDK*. https://www.ptgrey.com/ladybug-sdk, 2018. – Besucht: 31.07.2018

[Lot16] Lotz, Oliver: *Gefahren durch missbräuchliche Nutzung ziviler UAS aus Sicht der Polizei*. 16.03.2016. – Counter-UAS Kolloquium

[MMP12] Martinez-Martin, Ester ; Pobil, Angel P. d.: *Robust Motion Detection in Real-Life Scenarios* -. Berlin Heidelberg : Springer Science und Business Media, 2012. – ISBN 978–1–447–14216–4

[neu18] *Funktionsweise und Aufbau künstlicher neuronaler Netze*. "https://jaai.de/kuenstliche-neuronale-netze-aufbau-funktion-291/, 2018. – Besucht: 08.08.2018

[ope] *Eroding and Dilating*. https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html

[ope18a] *CUDA*. "https://opencv.org/platforms/cuda.html, 2018. – Besucht:31.07.2018

[ope18b] *OpenCV*. "https://opencv.org/, 2018. – Besucht: 31.07.2018

[ope18c] *Reading and Writing Video*. "https://docs.opencv.org/3.0-beta/modules/videoio/doc/reading_and_writing_video.html, 2018. – Besucht: 27.08.2018

[ope18d] *Smoothing Images*. "https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html, 2018. – Besucht: 13.08.2018

[qch18] *QCheckBox Class*. "http://doc.qt.io/archives/qt-4.8/qcheckbox.html, 2018. – Besucht: 27.08.2018

[qmu18] *QMutex Class*. "http://doc.qt.io/qt-5/qmutex.html, 2018. – Besucht: 29.08.2018

[qse18] *QSemaphore Class*. "http://doc.qt.io/archives/qt-4.8/qsemaphore.html, 2018. – Besucht: 03.08.2018

[qsi18] *Signal and Slots*. "http://doc.qt.io/archives/qt-4.8/signalsandslots.html, 2018. – Besucht: 29.08.2018

[qt18] *Qt*. "https://www.qt.io, 2018. – Besucht: 03.08.2018

[qtp18] *Einführung in Qt*. "http://www.mathematik.uni-ulm.de/sai/ws02/cpp/070103/Qt.pdf, 2018. – Besucht: 03.08.2018

[qtr18] *Threading Basics*. "http://doc.qt.io/qt-5/thread-basics.html, 2018. – Besucht: 03.08.2018

[qwi18] *Widgets and Layouts*. "http://doc.qt.io/archives/qt-4.8/widgets-and-layouts.html, 2018. – Besucht: 29.08.2018

[rin18] *Implementing a Queue using a circular array*. "http://www.mathcs.emory.edu/~cheung/Courses/171/Syllabus/8-List/array-queue2.html, 2018. – Besucht: 29.08.2018

[SA85] Suzuki, Satoshi ; Abe, Keiichi: Topological structural analysis of digitized binary images by border following. In: *Computer Vision, Graphics, and Image Processing* 30 (1985), Nr. 1, 32-46. http://dblp.uni-trier.de/db/journals/cvgip/cvgip30.html#SuzukiA85

[sch18] *UAV Schwarmintelligenz*. http://www.asctec.de/uav-uas-drohnen-anwendungen/schwarm-intelligenz/, 2018. – Besucht: 31.07.2018

[sie18a] *Siemensstern*. http://ps.bwii.at/topic/grafik/moire/siemens_star.html, 2018. – Besucht: 17.09.2018

[sie18b] *Siemensstern - Wie nutzt man ihn richtig?* "https://www.frag-den-neudeck.de/Archive/408, 2018. – Besucht: 30.08.2018

[smo18] *Bildglättung mittels Mittelwertfilter, Gaußfilter und Medianfilter*. "https://quisl.wordpress.com/2017/02/18/bildglaettung-mit-mittelwert-und-gaussfilter-faltung/, 2018. – Besucht: 13.08.2018

[SR14] Süsse, Herbert ; Rodner, Erik: *Bildverarbeitung und Objekterkennung - Computer Vision in Industrie und Medizin*. Berlin Heidelberg New York : Springer-Verlag, 2014. – ISBN 978–3–834–82606–0

[Ste08] Steinmüller, Johannes: *Bildanalyse - Von der Bildverarbeitung zur räumlichen Interpretation von Bildern*. Berlin Heidelberg New York : Springer-Verlag, 2008. – ISBN 978–3–540–79743–2

[Str97] Strecker, Stefan: *Künstliche Neuronale Netze - Aufbau und Funktionsweise*. 1997

[Sul17] Sullivan, Frost: *Global Defense Counter-UAS Technologies Market Forecast to 2022*. 01.10.2017. – Counter-UAS Kolloquium

[tcp18a] *SPECIFICATION OF INTERNET TRANSMISSION CONTROL PROGRAM*. "http://www.ietf.org/rfc/rfc0675.txt, 2018. – Besucht: 01.08.2018

[tcp18b] *Was ist TCP/IP*. "https://www.searchnetworking.de/definition/TCP-IP-Transmission-Control-Protocol-Internet-Protocol, 2018. – Besucht: 01.08.2018

[usa18] *Das USAF 1951 Target: Bestimmung und Berechnen des optischen Auflösungsvermögens eines Bildverarbeitungssystems*. "http://www.vision-doctor.com/optik-berechnungen/usaf-1951-bestimmung-aufloesung.html, 2018. – Besucht: 30.08.2018

A Anhang

Messung	LadybugCaptureThread	ImageThresholdThread	ImageProcessingThread	DetectionLogicThread
1	110	305	316	6
2	132	249	142	1
3	122	256	185	1
4	216	233	189	1
5	227	224	212	27
6	280	224	221	1
7	216	257	185	46
8	283	233	186	1
9	199	218	139	8
10	204	276	226	8
11	193	235	192	8
12	198	224	119	1
13	262	268	220	8
14	236	212	186	1
15	225	220	121	46
16	236	229	216	1
17	237	274	182	2
18	135	223	123	46
19	252	270	148	1
20	270	271	164	1
21	209	273	150	1
22	229	259	139	1
23	154	236	143	6
24	234	254	211	6
25	181	219	120	61
26	274	268	132	2
27	201	235	151	3
28	160	260	208	5
29	218	215	120	46
30	275	279	162	1
31	274	261	149	1
32	183	259	158	1
33	249	267	145	1
34	133	254	148	47
35	178	261	152	26
36	133	261	164	1
37	243	253	140	1
38	217	241	210	1
39	157	257	245	28
40	263	210	176	20
41	189	225	95	0
42	283	268	145	3
43	245	308	148	1
44	228	224	211	1
45	136	258	120	1
46	229	207	150	46
47	249	267	168	20
48	228	261	255	1
49	237	257	150	1
50	197	257	105	1
Mittelwert	212,38	249,1	168,84	10,98

Abbildung A.1: Messung der Dauer der einzelnen Threads in ms

B Anhang

Messung	Thresholding	Erosion	Dilation	Glättung
1	137	30	32	256
2	129	17	6	102
3	141	13	6	116
4	140	6	11	208
5	109	29	10	127
6	110	45	6	111
7	140	53	13	132
8	156	34	6	116
9	141	6	6	109
10	125	23	6	117
11	141	21	15	137
12	125	10	6	125
13	156	25	6	128
14	110	5	6	149
15	141	9	5	136
16	156	8	6	116
17	172	10	5	141
18	125	6	6	114
19	125	23	6	118
20	140	9	6	128
21	140	8	6	122
22	94	8	6	138
23	94	34	7	105
24	156	8	6	109
25	124	8	6	123
26	156	8	6	105
27	140	9	6	136
28	141	27	31	113
29	125	8	6	120
30	140	16	6	115
31	125	31	6	132
32	156	32	6	112
33	129	16	15	107
34	141	16	15	136
35	140	70	32	140
36	140	16	15	109
37	110	15	6	110
38	140	15	6	150
39	156	32	6	135
40	141	16	32	114
41	125	16	6	122
42	141	47	6	135
43	125	47	31	119
44	156	0	14	114
45	110	31	16	118
46	141	16	33	134
47	156	16	15	122
48	172	16	32	100
49	125	15	13	124
50	125	16	16	105
Mittelwert	135,66	19,9	11,52	126,2

Abbildung B.1: Messung der Dauer der einzelnen Bildverarbeitungsalgorithmen in ms