DIPLOMARBEIT

Autonome Flugdrohne

Evaluierung möglicher Systeme und praktische Implementierung einer Hinderniserkennung

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. Dietmar Dietrich
Univ.Ass. Dipl.-Ing. Martin Pongratz, BSc

am

Institut für Computertechnik (E384)
der Technischen Universität Wien

durch

Christoph Löffler
Matr.Nr. 0728205
Veronikagasse 41/20, 1170 Wien

Wien, am 01.10.2014 _____

Kurzfassung

In dieser Diplomarbeit soll ein System zur Hinderniserkennung zur Kollisionsvermeidung implementiert werden. Dies soll als Erweiterung für den bereits am Institut für Computertechnik entwickelten Hexacopter dienen. Zu diesem Zweck werden einige, dem derzeitigen Stand der Technik entsprechenden Hinderniserkennungssysteme, sowie deren Vor- und Nachteile evaluiert. Einige Systeme, welche in dieser Arbeit behandelt werden, sind z. B. Laserscanner, Ultraschall sowie auch Stereo- und Mono-Kamera-Systeme. Die Evaluierung umfasst die Hardwareaspekte und ihre möglichen Anwendungsbereiche, sowie die Verarbeitung der gewonnen Daten.
Für die Implementierung des Systems wurde ein Mono-Kamera-System in Kombination mit dem Optischen Fluss Algorithmus ausgewählt. Zu Beginn der Arbeit wurde versucht, die Umsetzung auf der bereits vorhandenen Hardware, einem Gumstix Overo Fire, in Verbindung mit einem CASPA Kamera Modul umzusetzen. Während der Umsetzung hat sich jedoch herausgestellt, dass der Gumstix Overo Fire die hohen Ansprüche des Algorithmus an die Hardware nicht erfüllen konnte. Durch diesen Umstand ist es notwendig geworden, eine leistungsstärkere Hardware zu finden, welche die gestellten Anforderungen erfüllen kann. Es stellte sich heraus, dass sich ein aktuelles Smartphone, welches über einen Qualcomm Chipsatz verfügt, sehr gut dafür eignet. OpenCV, sowie die optimierte Qualcomm Vision Library FastCV, wurden ebenfalls verwendet. Der Vorteil dadurch ist, dass man durch die Verwendung von OpenCV eine große Auswahl an Algorithmen hat. Weiters ermöglicht FastCV eine höhere Ausführungsgeschwindigkeit bei rechenintensiven Algorithmen, da diese speziell für den Qualcomm Chip optimiert wurde. Die Implementierung der Hinderniserkennung wurde sowohl mit synthetischen Szenen, als auch mit realen Szenen durchgeführt. Die Ergebnisse des implementierten Systems zeigten jedoch, dass die erhaltenen Ergebnisse nicht genau genug sind, um das System für eine zuverlässige Hinderniserkennung einzusetzen. Ein Grund dafür ist, dass der Multikopter wackelt und dadurch die Berechnung des Focus of Expansion Fehler aufweist, wodurch die Position des Focus of Expansion instabil ist. Dies hat einen großen Einfluss auf die Genauigkeit und auf das Endergebnis der Hinderniserkennung. Wird eine robustere Implementation des Systems realisiert, kann dies, gerade für Bodenfahrzeuge, zu guten Ergebnissen führen.

Abstract

This thesis deals about the implementation of an object detection system with the purpose to avoid collisions. The system is planed to be used by the already existing Hexacopter system of the Institute of Computer Technology at the Vienna University of Technology. Therefore some state of the art object detection systems and their assets and drawbacks are evaluated. These systems are laser scanner, ultrasonic and vision based systems like stereo and mono camera systems. The evaluation covers hardware aspects of the systems and possible usage and processing of the gathered data is also discussed. For the implementation of the object detection system a mono vision system in combination with optical flow estimation is chosen. The first approach was to handle the task with a Gumstix Overo Fire and a CASPA Camera Module. Unfortunately the Gumstix Overo Fire couldn't meet the computational high demands of the optical flow estimation. For this reason some other embedded systems had to be evaluated. The final choice was to use an up to date Smartphone with a Qualcomm Chipset and the optimized Qualcomm Vision library FastCV. OpenCV was used as well which can run concurrently with FastCV. This has the advantage of using numerous algorithms from OpenCV in combination with the optimized ones from FastCV. The implementation of the object detection system was evaluated with synthetic video sequences as well as in a real world environment. The evaluation of the implemented system indicates that the obtained measurements are not reliable enough to use the system for collision avoidance. One reason is the (unfirm) movement of a multicopter and therefore a unstable focus of expansion location which has a huge impact on the final result of the collision avoidance. If a more robust implementation of the system is realized this could lead to good results, especially for ground vehicles.

Danksagung

An dieser Stelle möchte ich mich bei all jenen Personen bedanken, die mich während meines Studiums und dem Schreiben meiner Diplomarbeit unterstützt haben. Besonderer Dank gilt hierbei meinen Eltern, für ihre Geduld, ihren Rat und ihren Beistand während meiner gesamten Studiendauer. Auch möchte ich mich bei meiner Freundin Barbara für ihre Unterstützung bedanken. Nicht zuletzt danke ich auch meinen Betreuern, Univ.Ass. Dipl.-Ing. Martin Pongratz, BSc sowie O.Univ.Prof. Dipl.-Ing. Dr.techn. Dietmar Dietrich, für ihre geduldige und engagierte Betreuung.

Inhaltsverzeichnis

Abkürzungen

2D	2 Dimensional
3D	3 Dimensional
3G	Third-generation
6DOF	Six degrees of freedom
Akku	Akkumulator
API	Application programming interface
AR	Augmented Reality
COM	Computer On Module
CPU	Central Processing Unit
DSP	Digital signal processor
EEPROM	Electrically Erasable Programmable Read-Only Memory
FOC	Focus of Contraction
FoE	Focus of Expansion
FOV	Field of View
FPS	Frames per second
FPV	First Person View
GPS	Global Positioning System
GPU	Graphics processing unit
I2C	Inter-Integrated Circuit
ICT	Institute of Computer Technology
IO	Input/Output
IR	Infrared
ITX	Integrated Technology Extended
JNI	Java Native Interface
LIDAR	Light detection and ranging
LiPo	Lithium-Polymer
LTE	Long Term Evolution
MEMS	Micro-electro-mechanical systems
MHz	Megahertz
OMAP	Open Multimedia Application Platform
PCL	Point Cloud Library
RADAR	Radio Detection and Ranging
RAM	Random-Access Memory
RANSAC	Random sample consensus
RC	Remote Controlled
RGB	RotGrünBlau
SDK	Software Developement Kit
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
TI	Texas Instruments

TTC	Time to Contact
UAV	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
UWB	Ultra-wideband
VLIB	Video Analytics und Vision Library
XDAIS	eXpress DSP Algorithm Interoperability Standard

1 Einleitung

Eine Flugdrohne ist ein unbemanntes Fluggerät welches in verschiedensten technischen Ausführungen gebaut und verwendet wird. Sie werden auch oft Unmanned Aerial Vehicles (UAV) genannt. Die Drohnen selbst sind selten komplett autonom. Größere Drohnen werden von Bedienpersonal, zum Teil von mehreren Personen, am Boden gesteuert. Kleinere Drohnen hingegen werden über eine Funkfernsteuerung bedient.

Der Unterschied zwischen einer normalen Drohne und einer Autonomen Drohne soll kurz aufgezeigt werden. Eine Drohne kann man als autonom bezeichnen, wenn sie selbstständig, ohne Hilfe einer agierenden Person, Aktionen ausführen kann. Diese Aktionen können unter anderem sein: vordefinierte Routen ab zufliegen, Hindernisse zu erkennen und diesen auszuweichen, selbständig zu landen und vieles mehr. Dies ist jedoch mit erheblichem technischen Aufwand verbunden. In dieser Arbeit soll beleuchtet werden, welche technischen Systeme dem Stand der Technik entsprechen. Außerdem wird ein System zur Hinderniserkennung umgesetzt.

Drohnen bekommen in der heutigen Zeit immer größere Bedeutung und durch den technischen Fortschritt erhöht sich der Einsatzbereich deutlich. Je nach Art der Aufgabe können verschiedene Drohnen eingesetzt werden. Sollen größere Strecken mit hoher Nutzlast überwunden werden, eignen sich Flächenmodelle sehr gut. Ist jedoch Agilität, in der Luft schweben und senkrechtes Starten und Landen nötig, kann auf Hubschrauber bzw. Multikopter zurückgegriffen werden.

Flächenmodelle erreichen hohe Geschwindigkeiten und werden u.a. zur Kartografierung, Aufklärung oder auch für Luftbildaufnahmen von großen Gebieten eingesetzt. Größere Ausführungen sind meist ähnlich wie Passagierflugzeuge aufgebaut und besitzen drei Arten von Rudern: Höhen-, Seiten- sowie Querruder, während Nurflügel Modellen nur Höhen- und Querruder zur Verfügung stehen. Flächenmodelle werden zum Beispiel vom US Militär eingesetzt. Ein Beispiel dafür wäre die, für militärische Anwendungen genutzte, Predator oder Reaper Drohne mit Flügelspannweiten bis zu 20 Metern. Auch das Österreichische Bundesheer setzt mittlerweile auf den Einsatz von Drohnen [55]. Nurflügel Modelle werden gerne hobbymäßig verwendet, unter anderem zum First Person View (FPV) fliegen. Sie haben den Vorteil, dass sie relativ leicht, robust und schnell sind. Anders als bei Multikoptern steckt weniger technischer Aufwand dahinter, das bedeutet, dass selbst bei einem Absturz weniger kaputt gehen kann, was eine Kosteneinsparung zur Folge hat.

Um Drohnen sicher autonom steuern zu können, ist es notwendig, dass ein Steueralgorithmus über seine Umgebung Bescheid weiß. Wichtig ist es hier zu wissen,in welche Richtung sich die Drohne bewegt, wie sie ausgerichtet ist, ihre Flughöhe und ob Hindernisse auf dem Flugpfad zu erwarten sind. Diese Informationen dienen dazu neue Steuerungsbefehle zu errechnen und dadurch einen

stabilen, sicheren Flug ohne Abstürze zu gewährleisten. Um Hindernisse zu erkennen gibt es eine Vielzahl an verschiedenen Sensoren, die alle Vor- und Nachteile mit sich bringen.

Anforderungen an die Systeme sind zum Beispiel:

- Zuverlässige Erkennung der Umgebung
- Schnelle Erkennung der Systeme
- Funktion im Freien (auch bei starkem Sonnenschein) gegeben
- Leistungsverbrauch
- Kosten
- Gewicht

In Kapitel 2 werden einige aktuelle Systeme zur Hinderniserkennung näher betrachtet. Ein System, dass die Anforderungen erfüllt oder ihnen zumindest am ehesten entspricht, wird anschließend ausgewählt. Es sind jedoch nicht nur die Sensoren relevant. Wichtig ist auch zu diskutieren, wie die gewonnenen Informationen in kurzer Zeit verarbeitet und wenn nötig auch aufbereitet werden können.

1.1 Überblick über am Markt befindlicher Drohnen

Da es unzählige verschiedene Arten von Drohnen gibt sollen die bereits zuvor angesprochenen Drohnen etwas genauer betrachtet werden. Dazu werden diese in drei Kategorien eingeteilt: Flächenmodelle, Hubschrauber, sowie Multikopter mit ihren verschiedenen Variationen. Jede Kategorie hat ihre Stärken und Schwächen und dadurch ergeben sich auch zum Teil verschiedene Anwendungsgebiete. Vor allem Multikopter erfreuen sich in den letzten Jahren an immer größerer Beliebtheit. Der Preisverfall der benötigten Komponenten sowie die erhöhte Leistungssteigerung von Microcontrollern bzw. Microcomputern hat einen nicht unerheblichen Anteil an diesem Erfolg welcher auch kommerziell gut vermarktet wird. Mittlerweile gibt es bereits Geräte wie die AR.Drone [4] welche als Spielzeug eingesetzt und über das Mobiltelefon gesteuert werden kann.

1.1.1 Flächenmodelle

Flächenmodelle sind sehr beliebt wenn größere Strecken bewältigt oder größere Nutzlasten transportiert werden müssen. Durch ihre aerodynamische Bauweise und der Flügel profitieren sie vom Auftrieb. Wodurch sie mit weniger Leistung bzw. Energieverbrauch größere Strecken bewältigen können und/oder höhere Nutzlast transportieren.

Beim Österreichischen Bundesheer werden die Drohnen unter anderem zur Raumüberwachung bei Auslandseinsätzen oder zur Aufklärung nach Naturkatastrophen im Inland eingesetzt. [55]. Diese Drohnen werden jedoch vom Bedienpersonal am Boden gesteuert und sind nicht autonom. Während die Aufklärung und die Begutachtung von Naturkatastrophen mithilfe einer Drohne sehr sinnvoll erscheint, ist ihr Kampfeinsatz nicht unbedingt gut zu heißen. Da jedoch gerade für

militärische Anwendungen sehr viel Geld in die Forschung und Entwicklung investiert wird, muss mit einer Weiterentwicklung in diese Richtung gerechnet werden.

Ein anderes Anwendungsgebiet sind Hobbypiloten, die Flächenmodelle wie Segelflugzeuge, motorbetriebene Flugzeuge oder auch Nurflügler steuern. Diese Modelle werden via Funkfernsteuerung betrieben und können somit im Sichtbereich des Piloten geflogen werden. Eine Erweiterung dazu stellt der sogenannte First Person View (FPV) Flug dar. Dabei wird auf der Drohne eine Kamera montiert. Über ein Funksystem wird das Videobild der Kamera zu einer Videobrille des Piloten übertragen und somit kann die Drohne auch außerhalb des Sichtbereichs des Piloten gesteuert werden. Diese Modelle benötigen im Vergleich zu Multikoptern keine Lagestabilisierung und besitzen deshalb meist nur Motore(n) und Servos, jedoch keinen extra Microcontroller der die Aufgabe der Lagestabilisierung übernimmt. Ansich wäre es jedoch einfach einen Microcontroller in solchen Modellen einzubauen und diese zum Beispiel autonom vordefinierte GPS Routen abfliegen zu lassen.

1.1.2 Hubschrauber

Hubschrauber lassen sich genauso wie Flächenmodelle ebenfalls wieder zwei Anwendungsgebiete unterteilen, einerseits in Drohnen für militärische Anwendungen und andererseits als Freizeitbeschäftigung. Hubschrauber oder auch Helikopter sind technisch bereits sehr ausgereift da sie nicht nur in Modellen bzw. Drohnen zum Einsatz kommen, sondern auch in der zivilen Luftfahrt seit langem in Verwendung sind.

Ein Beispiel für eine Militärische Drohne wäre der Hummingbird A160 von Boing (Abbilung 1.1 oder auch der CamCopter von der österreichischen Firma Schiebel. Bei den Hobbyausführungen ist zwischen günstigen Modellen ab ca. 20 € wie in Abbildung 1.2 (a) und den eher höherpreisigen zu unterscheiden. Die günstigen Modelle können sich meist nicht seitwärts bewegen, wodurch nicht das Flugverhalten eines richtigen Helikopters erreicht wird. Ein gutes Helikopter Modell wäre z. B. ein Modell aus der T-Rex Serie, wie in Abbildung 1.2 (b) zu sehen. Dieses schlägt jedoch mit Kosten von mehreren hundert Euro zu Buche.

Abbildung 1.1: Boing Hummingbird A160 [7]

(a) Syma S107 Helikopter. Abbildung von Sayamindu Dasgupta unter CC BY-SA 2.0 [51]

(b) T Rex 450 se v2. Abbildung von andy_c unter CC BY 2.0 [52]

Abbildung 1.2: Modellhelikopter

1.1.3 Multikopter

Die bekanntesten Multikopter Ausführungen sind Trikopter, Quadrokopter, Hexakopter sowie Octokopter. Der Name deutet bereits auf die Anzahl an Auslegern bzw. auf die Anzahl der verbauten Motoren hin. Am Ende jedes Auslegers wird ein Motor befestigt. Ein Trikopter besitzt drei, ein Quadrokopter vier, ein Hexakopter sechs und ein Octokopter acht Motoren. Die Anzahl der Motoren ist nicht nur für die maximal zu transportierende Nutzlast, sondern auch für die Ausfallsicherheit wichtig. Während ein Tri- und Quadrokopter bei Ausfall eines Motors abstürzen so könnten Hexa- und Octokopter dies kompensieren und so einen Absturz verhindern.

Multikopter sind meist sehr ähnlich aufgebaut. Sie bestehen aus einer Grundplatte im Zentrum des Gerätes auf der sich die nötige Technik sowie der Akku befinden. An der Grundplatine ist ein Landegestell sowie je nach Multikopter Ausführung die Anzahl an Auslegern, an denen die Motoren befestigt sind, angebracht. Als Material für die Grundplatine und die Ausleger kommt bei den günstigen Modellen oft Plastik oder Aluminium zum Einsatz während bei den preislich höher angesiedelten Modellen auf Karbon zurückgegriffen wird. Dies wird vor allem aus Gründen der Gewichtseinsparung gemacht, da dies zu einigen Vorteilen führt. Diese wären unter anderem:

- Längere Flugdauer mit einer Akkuladung
- Schwerere Ladung kann transportiert werden
- Bessere Flugeigenschaften z. B. agiler und bessere Beschleunigung

Auf der zentralen Grundplatte befindet sich die Steuereinheit sowie zusätzliche Sensoren. Typisch sind für die Steuereinheit etwa ein Arduino mit ATMega Chip oder ein etwas leistungsfähiger ARM Chip. Dies hat den folgenden Grund: um einen stabilen Flug des Hexakopters zu gewährleisten ist es notwendig, dass die Regelschleife entsprechend schnell ist um Sensordaten auszulesen, zu verarbeiten und anschließend Steuerungsbefehle zu senden.

Als zusätzliche Ausstattung sind oft auch noch Luftdrucksensoren und GPS im Einsatz um sicheres Landen,Positionshalten (an der Stelle schweben) oder auch "Comming Home“ (automatischer Rückflug zum Startplatz via GPS Steuerung) zu gewährleisten. Zusätzlich zu den oben genannten

Sensoren sind noch Motorcontroller, RC Empfänger und ein LiPo Akku zur Stromversorgung, an Board.

1.2 ICT Hexakopter

Die am Institut für Computertechnik von Peter Hanger [PH12] entwickelte Drohne wie in Abbildung 1.3 zu sehen, ein Hexakopter, ist die technische Basis dieser Diplomarbeit. In den nachfolgenden Unterkapiteln wird der derzeitige Stand der via Funkfernbedienung steuerbaren Drohne sowie die nötigen Erweiterungen für ihr autonomes Agieren, aufgezeigt.

Abbildung 1.3: Am ICT entwickelter Hexacopter von Peter Hanger

1.2.1 Ausstattung des Hexakopter

Der Hexakopter besteht wie für einen Hexakopter üblich aus sechs Auslegern mit einem Durchmesser von ca. 86 cm. Als Bausatz für das Grundgerüst wurde auf einen Bausatz von Mikrokopter zurückgegriffen.

In Tabelle 1.1 sieht man einen Auszug der Kosten des Verbauten Computersystems und Sensoren. In Summe sind Gesamtkosten von etwa 900 Euro entstanden, wobei davon ca. 230 Euro auf das Funkfernsteuerungssystem entfallen.

Ebenfalls wurde bereits eine Kamera, nämlich das Gumstix Caspa Kamera Modul erworben. Dieses Modul kann über ein 27 Pin Flex Kabel direkt mit dem Gumstix Overo verbunden werden. Die Kamera wurde jedoch noch nicht verbaut.

Bezeichnung	Beschreibung	Kosten
Gumstix Overo Fire COM	OMAP3530 Prozessor mit DSP	159,45 €
Breakout Board mit dem BMP085 absoluten Drucksensor	BMP085 - Präziser Low Power Drucksensor	17,26 €
9 Degrees of Freedom - Sensor Stick	Der Stick besteht aus einem ADXL345 accelerometer, HMC5883L magnetometer und einem ITG-3200 MEMS gyro	72 €
AVR32 Modul mit EEPROM	AVR32-Entwicklungsmodul	34,90 €
Sharp GP2Y0A02YK IR Entfernungssensor	IR Entfernungssensor mit einem Messbereich von 20-150cm	11,22 €

Tabelle 1.1: Auszug der Kosten für das Computersystem sowie Sensoren

Für eine genaue Übersicht der verbauten Komponenten, Kosten und der weiteren Ausstattungsmerkmale des am Institut für Computertechnik eingesetzten Hexakopters, wie u.a. die verwendeten Motoren, Rahmenset und Brushless Motor Controller sei auf [PH12, S. 120] verwiesen.

Wie der Tabelle zu entnehmen ist, besitzt der Hexakopter nicht nur einen ATmel 32 Bit Microcontroller sondern auch einen Gumstix Microcomputer mit Embedded Linux. Der Microcontroller dient zur Lagestabilisierung. Es werden dabei die Sensordaten von Gyrometer, Kompass und Accelerometer verarbeitet und durch Sensorfusion vereint. Die Daten werden in einer Regelschleife ausgewertet und neue Parameter für die Lagestabilisierung, die an die Motorcontroller gesendet werden, berechnet. Die Lagestabilisierung ist für einen Multikopter, also auch für den hier verwendeten Hexakopter, zwingend erforderlich da diese nicht eigenstabil wie etwa Flächemodelle sind und bei Versagen der Regelschleife sofort abstürzen würden.

Die Anforderung an die Regelschleife kann ein Microcomputer, wie der Gumstix Overo auf dem ein Embedded Linux Betriebssystem läuft, selbst mit einer Echtzeiterweiterung wie Xenomai, nicht ohne erheblichen Aufwand sicherstellen [PH12, S. 52].

Ein Microcomputer ist dennoch nützlich und kann für vielerlei Aufgaben verwendet werden, um nur einige wenige davon anzuführen:

- Kamera anschließen um Videos oder Fotos aufzunehmen
- Videostream an ein Empfangsgerät senden um auch außer Sichtkontakt die Drohne fliegen zu können z. B. FPV Flüge
- Verwendung von GPS Mäusen um vorprogrammierte Routen abzufliegen oder auch nur zum Aufzeichnen von Routen
- Anschließen eines UMTS Modules um auch außerhalb der Reichweite der Funkfernbedienung die Drohne steuern zu können.

1.2.2 Autonome Funktion

Derzeit kann der Hexakopter ausschließlich mit einer MX-16S Funkfernsteuerung geflogen werden. Die Funkfernsteuerung kommuniziert über das 35 MHZ Band mit dem Empfänger der auf der Drohne verbaut ist und kann so Steuerbefehle übertragen.

Die Drohne kann manuell geflogen werden, aber es fehlen noch einige weitere wichtige Funktionen um sie autonom agieren zu lassen. Ein Ziel dieser Arbeit ist es, die vorhandene Drohne autonom navigieren zu lassen. Dazu ist es notwendig, dass die Umgebung sozusagen analysiert und daraus Steuerungsbefehle errechnet werden können. Die derzeit verbauten Sensoren allein reichen dafür noch nicht aus.

Es gibt verschiedene Ansätze wie man eine autonome Drohne steuern kann. Ein Ansatz wäre, wie in [Mul], dass die Drohnen sich in einem geschlossenen Raum befinden. In diesem Raum sind hochauflösende Kameras so positioniert, dass der komplette Raum erfasst werden kann. Die somit gewonnen Bilddaten werden an ein Rechnersystem weitergeleitet. Dieses Rechnersystem wertet die gewonnenen Daten aus und kann damit die Position der Drohne und auch andere im Raum befindliche Objekte erkennen. Daraus kann dann die Flugbahn berechnet werden. Für ein Kunstprojekt [Wil] wurde dies durchgeführt. Ein Beispiel kann man in Abbildung 1.4 betrachten. Der Computer hat sämtliche Planung über Flugbahn, Akkunachladezeiten, Positionierung der einzelnen Blöcke etc. übernommen und hat nur die Steuerbefehle an die Drohne gesendet. In solch einem Szenario ist kann man die Drohne zwar als autonom bezeichnen da sie nicht von einem Menschen gesteuert wird, jedoch ist sie weiterhin von einem Rechnersystem das Steuerbefehle sendet abhängig. Sie ist also nicht völlig autark wie es die Drohne in dieser Diplomarbeit sein soll.

Abbildung 1.4: Flight Assembled Architecture installation (Gramazio & Kohler and Raffaello D'Andrea in cooperation with ETH Zurich, Abbildung: © François Lauginie)

Ein anderer Ansatz wäre, dass man die Berechnung direkt durch ein Computersystem welches auf der Drohne verbaut ist durchführt und nicht erst die Daten an ein Rechensystem überträgt. Dazu könnte z. B. der derzeit auf der Drohne verbaute Gumstix Computer dienen. Dieser hat im Gegensatz zu einem handelsüblichen Computersystem, welches zum Beispiel in der Arbeit [FFP12] zum Einsatz kam, einige Vorteile:

- niedriges Gewicht
- kleines Ausmaß
- relativ leistungsfähig
- geringer Stromverbrauch
- viele IO Ports, nicht nur USB, für diverse Erweiterungen und Sensoren vorhanden

Da die Drohne relativ groß und primär für Flüge im freien Raum ausgelegt ist, wird diese Arbeit sich mit dem 2. Ansatz, also einem Computersystem direkt auf der Drohne, befassen.

1.3 Einsatzgebiete von Autonomen Drohnen

Einige Beispiele an Aufgaben welche Drohnen derzeit schon im Stande sind zu erledigen, jedoch nicht autonom, werden in nachfolgend aufgeführt. Der Wunsch geht in die Richtung, dass Drohnen dies ohne Hilfe von Bedienpersonal erledigen können oder zumindest das Bedienpersonal bei eintönigen und anstrengenden Arbeiten entlasten.

- Kartografierung (SLAM/LIDAR)
  Für die Kartografierung eigenen sich Drohnen sehr gut. An ihnen kann ein SLAM/LIDAR System angebracht werden das geologische Daten bei einem Überflug sammelt und welche später ausgewertet und zu einer Karte vereint werden können.
- Luftbildaufnahmen bzw. Filmaufnahmen
  Drohnen sind im Vergleich zu Helikoptern sehr viel billiger in der Anschaffung und Erhaltung. Die Kombination aus gutem Pilot der die Drohne steuert und zusätzlicher Foto- bzw. Videokamera ergibt sehr gute Aufnahmen. Diese Kombination ersetzt immer öfter richtige Helikopter.
- Aufklärung/Überwachung
  Wie bereits zu Anfang erwähnt, werden Drohnen häufig, vor allem vom Militär, zur Aufklärung und Überwachung eingesetzt. Dabei wird jedoch immer noch auf Bedienpersonal zurückgegriffen.
- Katastrophenhilfe
  Nach Unwettern könnten die Drohnen helfen das Ausmaß der Verwüstung besser abzuschätzen um so schneller Maßnahmen treffen zu können. Eine andere Anwendung wäre z. B. für Lawinenverschüttete, hierfür wurde bereits eine Drohne entwickelt [56].

Große Unternehmen sind ebenfalls angetan von den Möglichkeiten der autonomen Drohnen, vor allem im Transportbereich. Amazon verfolgt dieses Ziel ebenfalls, Paketzustellung direkt per Drohnen zu ermöglichen, jedoch wird nicht mit einer Einführung vor 2015 gerechnet [1]. In Abbilung 1.5 ist der derzeitige Prototyp einer DHL Drohne zu sehen. Auch ein anderer großer Paketdienstleister UPS verfolgt diesen Ansatz ebenso. Abgesehen von den regulatorischen Hindernissen müssen auch die technischen Entwicklungen voranschreiten um ein Abstürzen der Drohne zu verhindern.

Abbildung 1.5: Test einer Drohne von DHL. Abbildung von Frank Höffner unter CC BY-SA 3.0 [14]

Ein Absturz einer Drohne wäre fatal. Nicht nur, dass die Drohne selbst schwer beschädigt werden würde, auch würde die Nutzlast (zum Beispiel eine teure Spiegelreflex Kamera) oder ähnliches in Mitleidenschaft gezogen werden. Dies wäre nur ein finanzieller Schaden, noch schlimmer wäre es, wenn die Drohne über Menschengruppen abstürzen würde und so Personen verletzt. Solche Szenarien müssen unbedingt verhindert werden.

2 Aktuelle Systeme zur Hinderniserkennung

Es gibt eine Vielzahl an verschiedenen Geräten, die zur Hinderniserkennung eingesetzt werden können. In den nachfolgenden Unterpunkten werden gängige Systeme betrachtet und ihre Vor- und Nachteile kritisch diskutiert. Außerdem wird ihre Eignung für den Einsatz auf dem ICT Hexacopter geprüft. Es zählt also nicht nur ihre Leistungsfähigkeit sondern durchaus auch ihre Vielseitigkeit, auch im Bezug auf die Erweiterung des ICT Hexacopters, da sich manche Systeme auch für mehr eigenen als nur für eine Hinderniserkennung. Diese Systeme haben natürlich hinsichtlich Erweiterbarkeit Vorteile. Jedes System hat seine Stärken und Schwächen. Es soll deshalb ein System ausgewählt werden, dass am besten die Anforderungen, die nach Kapitel 1 gestellt worden sind erfüllen kann.

Das Kapitel ist in zwei Unterkapitel aufgeteilt, einerseits in die Gruppe der Optischen Systeme und andererseits in die Gruppe der Akustischen Systeme. Dabei werden nicht nur die Hardware sondern auch existierende Methoden auf ihre Eignung für den Einsatz am ICT Hexacopter diskutiert.

2.1 Optische Systeme

Optische Systeme treten meist in zwei unterschiedlichen Ausführungen auf. Einerseits als Aktive Systeme wie z.B. Laser Scan Systeme und andererseits als passive Systeme wie Mono Kamera oder Stereo Kamera Systeme. Kombinierte Systeme sind jedoch auch möglich, wie die von Microsoft vertriebene Kinect zeigt. Diese vereint eine RGB und IR Kamera sowie eine Infrarot Lichtquelle.

Da für Stereo- und Mono-Kamera-Systeme das Lochkamera Modell von essentieller Bedeutung ist wird dieses sowie die Epipolargeometrie kurz vorgestellt bevor Laser Systeme sowie kombinierte Ansätze behandelt werden. Dabei wird jedoch nicht nur auf die Hardware sondern auch auf die Software, also auf Methoden mit welche man Hindernisse erkennen kann, eingegangen.

2.1.1 Lochkamera Modell

Über das Lochkamera Modell stellt man die Grundlagen für die mathematischen Beziehungen zwischen dem 3D Raum und die perspektivische Projektion des 2D Bildes vereinfacht dar. Das Lochkamera Modell ist eines der wichtigsten Modelle in der Bildverarbeitung. Die Zusammenhänge, Beziehungen und Funktionsweise sollen nachfolgend Kurz erläutert werden.

Durch die Lochblende gelangen die Lichtstrahlen eines Objektes ins Innere der Kamera wo sie ein auf den Kopf gestelltes, spiegelverkehrtes und skaliertes Abbild des Objektes auf die Bildebene projizieren. Vertauscht man nun die Ebene der Lochblende, auch Brennebene genannt, mit der Bildebene, erhält man den folgenden Zusammenhang:

$$\frac{x}{f} = \frac{X}{Z} \tag{2.1}$$

Wobei f der Brennweite entspricht, also der Länge von der Lochblende zur Bildebene, welche bei realen Kameras sozusagen dem Bildsensor entspricht. Z ist der Abstand von der Brennebene zum Objekt, X die Größe des Objektes und x die verkleinerte Abbildung des Objektes auf der Bildebene. Analog dazu verhält es sich für Y bzw. y. Dadurch ist das abgebildete Objekt auch nicht mehr auf den Kopf gestellt [Bra08]. Dieser Zusammenhang wird in Abbildung 2.1 veranschaulicht.

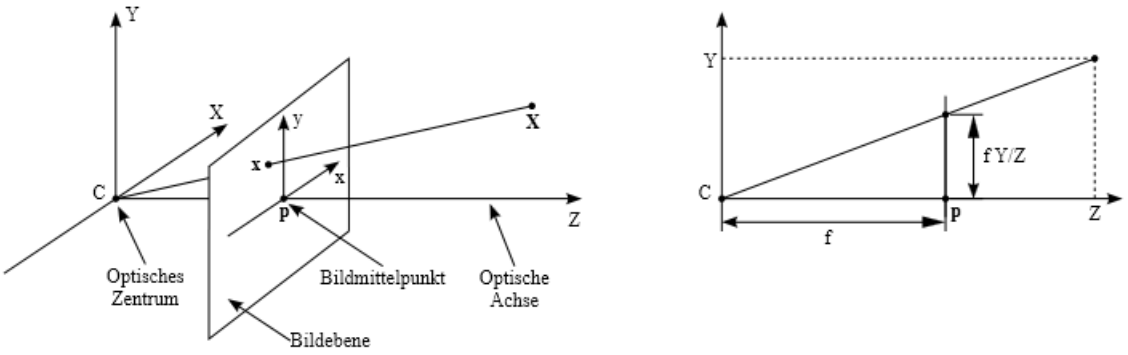


Abbildung 2.1: Lochkamera Modell bei dem die Bildebene vor das optische Zentrum gelegt wurde [HZ04]

Das optische Zentrum befindet sich jedoch nur beim idealen Modell exakt in der Mitte. Bei einer realen Kamera ist dies häufig nicht der Fall da beim Herstellungsprozess der Kamerasensor nicht immer so exakt ausgerichtet ist. Dies trifft oft vor allem auf günstig hergestellte Kameras zu, bei denen bei der Fertigungsqualität gespart wird. Um die Verschiebung des optischen Zentrums um die optische Achse auszugleichen kann man die Kamera kalibrieren.

Die Kamerakalibrierung ist ein häufiges Thema in der Wissenschaft und auch nicht unwichtig da selbst zwei Kameras vom selben Modell Abweichungen aufweisen können. Daraus folgt, dass die gewonnen Bildinformationen Fehler enthalten und so keine korrekte Darstellung der Szene möglich ist.

Man kann die Kamera vorab kalibrieren, z.B. mit der Camera Calibration Toolbox für Matlab oder auch OpenCV stellt dafür Funktionen zur Verfügung [38]. Die Kalibrierung basiert dabei auf dem Algorithmus von Zhang [Zha00]. Die Kalibrierung dauert mittlerweile weniger als eine Minute und man benötigt dazu kein extra Equipment das teuer angeschafft werden muss. Ein einfaches Schachbrettmuster, das man sich ausdrucken kann ist ausreichend dafür. Das Muster das Schachbrettes, genauer gesagt die Ecken der Quadrate, werden dabei als Kalibrierungsmarken benutzt. Als nächstes muss das Schachbrett aus verschiedenen Perspektiven aufgenommen werden. Wird die Kalibrierung über die von OpenCV zur Verfügung gestellten Funktionen durchgeführt

(bzw. genauer gesagt über das Kalibrierungs-Beispielprogramm) wird eine Mindestanzahl von zehn verschiedenen Perspektiven empfohlen. Umso mehr verschiedene Ansichten aus verschiedenen Perspektiven aufgenommen werden, umso genauer können die Kameraparameter bestimmt werden.

Die Methode von Zhang ist natürlich nicht die einzige Möglichkeit Kameras vorab zu kalibrieren. Eine etwas ältere Methode wäre z.B. die von Tsai [Tsa86] von 1986.

Es ist jedoch nicht nötig Kameras vorab zu kalibrieren. Es existieren auch Methoden die dies zur Laufzeit zulassen [MF92], [FLM92], [Har94]. Die Kalibrierung ist schwieriger durchzuführen als wenn man ein gegebenes Objekt mit dessen Ausmaßen kennt, wie z.B. das Schachbrettmuster. Doch nicht immer ist eine Vorab-Kalibrierung ausreichend oder es wird eine erneute Kalibrierung nötig. Das kann der Fall sein, wenn mechanische Einflüsse auf eine Kamera wirken oder z.B. im Falle eines Stereo Kamera Aufbaus eine Kamera verschieben. So kann durch die Selbstkalibrierung dem entgegen gewirkt werden.

Nach einer erfolgreichen Kalibrierung der Kamera erhält man die kameraspezifischen Parameter. Das sind einmal die bereits angesprochenen Abweichungen vom Mittelpunkt, dem optischen Zentrum, also px und py, sowie die Brennweite fx und fy. Wie man sieht ebenfalls für die x und y Koordinate. Die aus diesem Zusammenhang entstandene Matrix heißt Kamerakalibrierungsmatrix K. [HZ04]

$$K = \begin{bmatrix} fx & 0 & px \\ 0 & fy & py \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Durch die Kalibrierung werden jedoch nicht nur die Kameraparameter bestimmt sondern auch die Verzeichnung. Durch die Verwendung von qualitativ hochwertigen Objektiven kann die Verzeichnung minimiert oder auch ganz entfernt werden. Bei günstigen Kameras, wie z.B. Webcams die eine entsprechend nicht allzu hochwertige Optik verbaut haben ist jedoch mit Verzeichnung zu rechnen und sollte korrigiert werden. In Abbildung 2.2 ist die tonnenförmige sowie kissenförmige Verzeichnung abgebildet.

Abbildung 2.2: (Links) Quadrat ohne Verzeichnung. (Mitte) Tonnenförmige Verzeichnung, (Rechts) Kissenförmige Verzeichnung

2.1.2 Stereo-Kamera-Systeme

Stereo-Kamera-Systeme sind passive Systeme. Sie können also Informationen über ihre Umgebung sammeln ohne aktiv in den Erhalt der Informationen eingreifen zu müssen. Das bedeutet es müssen keine Lichtstrahlen ausgesendet werden wie es z.B. bei Laser Scannern nötig ist.

Häufig werden die zwei benötigten Kameramodule horizontal nebeneinander in einem nicht allzu großen Abstand aufgebaut, ähnlich wie das Augenpaar des Menschen. Dies hat mitunter zwei Gründe, einerseits ist die Größe auf einer Drohne durch ihre Maße begrenzt und andererseits bestimmt die Basislinie, also der Abstand zwischen den beiden Kameras, auch die Genauigkeit des Systems. Das bedeutet je nachdem wie die Länge der Basislinie gewählt wird, wird entweder die Genauigkeit im Nahbereich oder die im Fernbereich beeinträchtigt. Ansätze wie [GFMP08] und [OK93] zeigen jedoch wie man die Ungenauigkeiten im Nah- bzw. Fernbereich minimieren kann. Außerdem ist zu beachten, dass sich die beiden Bilder bei einer größeren Basislinie weniger überlappen, wodurch das Finden von Korrespondenzpunkten in beiden Bildern aufwendiger wird. Dies führt dazu, dass das allgemeine Ergebnis der Berechnung darunter leidet. Die ICT Drohne hat durch ihren großen Durchmesser genügend Optionen ein Stereo Kamera System zu positionieren und so den Messfehler im gewünschten Entfernungsbereich klein zu halten. Da die Hinderniserkennung nicht auf den Millimeter genau sein muss, stellen auch kleinere Ungenauigkeiten im Messvorgang wenig Probleme dar. Als guter Richtwert für die Länge der Basislinie dient der Augenabstand von rund 6 cm. Bei der Genauigkeit über große Distanzen können Stereo-Kamera-Systeme nicht mit Laser Systemen konkurrieren. Auch ist durch den Durchmesser der Nahbereich begrenzt, da die Entfernung von der Drohnenmitte bis zu den Spitzen der Ausleger mehr als 40 cm beträgt.

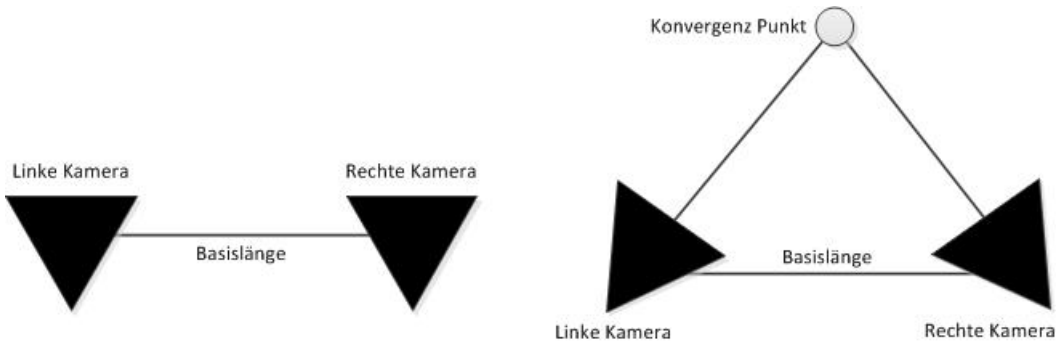


Abbildung 2.3: Achsparalleles Stereo System (links), konvergente Anordnung (rechts) nach [Sch05]

Bei der Ausrichtung der Kameras kann im Prinzip zwischen einer achsparallelen Ausrichtung Abbildung 2.3 (Links) und einer konvergenten Ausrichtung unterschieden werden Abbildung 2.3 (Rechts). Die achsparallele Ausrichtung hat den Nachteil, dass man mit ihr je nach Beschaffenheit der Szene keine Tiefeninformationen gewinnen kann. Um diesen Nachteil auszugleichen wird in Produktivsystemen die konvergente Anordnung gewählt und durch die sogenannte Rektifizierung in ein quasi achsparalleles Stereo System überführt. Dabei erfährt die zweite Kamera eine virtuelle Drehung um so wieder parallel ausgerichtet zu sein. Dadurch sind die Nachteile ausgeglichen und die Vorteile der einfacheren Berechnung die ein achsparalleles Stereo System bietet können ausgeschöpft werden.

Um Tiefeninformationen aus dem Bild zu extrahieren müssen zwischen den zwei Kameras Korrespondenzen gefunden werden. Durch die Rektifizierung kann der Suchbereich eingeschränkt werden was wiederum der Geschwindigkeit zu Gute kommt. Dadurch muss nur noch entlang der Epipolarlinien nach Korrespondenzen gesucht werden und nicht mehr im gesamten Bild. In Abbildung 2.4 (1) sieht man den Suchraum ohne Rektifizierung und in Abbildung 2.4 (2) mit Rektifizierung. Dabei werden die Bildebenen so gedreht, dass sie nach der Drehung die Eigenschaften eines Achsparallelen Stereo Systems haben. Wie man sieht ist dies ein sehr wichtiger Schritt bei

Stereo-Kamera-Systemen. Einige andere Methoden zur Triangulation wie z.B. die Polynomialmethode, die lineare und die iterative Methode werden in [HS97] diskutiert und verglichen.

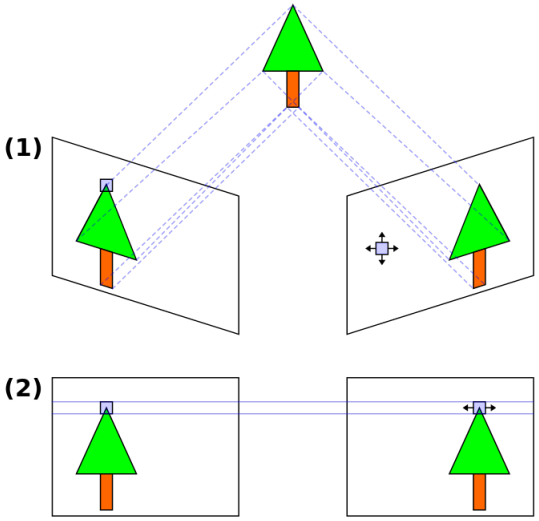


Abbildung 2.4: (1) ohne Rektifizierung, (2) mit Rektifizierung. Abbildung von Bart van Andel unter CC BY-SA 3.0 [27]

Das Korrespondenzproblem ist damit jedoch noch nicht komplett gelöst. Um gute Ergebnisse zu erzielen ist es sinnvoll die Kameras zu kalibrieren um u.a. die Verzeichnungen zu korrigieren, wie bereits im vorigen Unterpunkt diskutiert wurde. Aber das allein reicht noch nicht aus. Die Bildqualität darf nicht zu schlecht sein um genug Details der Szene erkennen zu können. Stimmen also die Lichtverhältnisse nicht, kann dies bei günstigen Kameras sehr schnell zu Problemen führen. Im Fall bzw. im Einsatzgebiet der ICT Drohne sind Lichtverhältnisse für Stereo-Kamera-Systeme als weniger kritisch zu erachten, da die Drohne durch ihre Größe primär für den Einsatz im freien Gelände konzipiert ist und somit ausreichend Licht zur Verfügung steht um eine ausreichend gute Bildqualität zu liefern. Solange man nicht im Dunkeln oder in der Dämmerung fliegt, sollte das Licht an einem normalen bewölkten oder auch sonnigen Tag kein Problem darstellen.

Ein Problem, das nicht so leicht zu beheben ist, sind Szeneninformationen. Man stelle sich vor die Drohne bewegt sich im Freien auf eine weiße, fast glatte Wand zu ohne dass diese weitere markante Bildmerkmale besitzt. Es wird daher sehr schwer den gefundenen Punkt von Kamera 1 auch in Kamera 2 zu finden, da alles gleich aussieht. Das Gleiche wäre z.B. bei einem Netz. Ein anderes Problem wäre wenn durch die zwei verschiedenen Ansichten der Stereo Kameras in einem Bild ein Objekt verdeckt ist, während es im anderen Bild sichtbar ist. Da das Objekt in einem Bild so gesehen nicht existiert kann auch keine Korrespondenz gefunden werden.

Hat man die Korrespondenzpunkte bestimmt, kann zur eigentlichen Berechnung der Tiefeninformationen und somit auch zur Hinderniserkennung übergegangen werden. Bei Stereo Kameras wird sehr oft das Prinzip der Triangulation verwendet um den Abstand zu einem Objekt zu bestimmen. Dieses Verfahren eignet sich aufgrund der bekannten Basislinie zwischen den zwei verwendeten Kameras sehr gut.

In [Sch05] wird davon ausgegangen, dass die Punktkorrespondenzen nicht absolut exakt sind was u.a. durch z.B. Bildrauschen verursacht werden kann. Deshalb wird als Approximation die kürzeste Strecke zwischen den zwei optischen Strahlen herangezogen. Dieser Zusammenhang ist in Abbildung 2.5 dargestellt, wobei die mit s bezeichnete Linie die kürzeste Strecke darstellt.

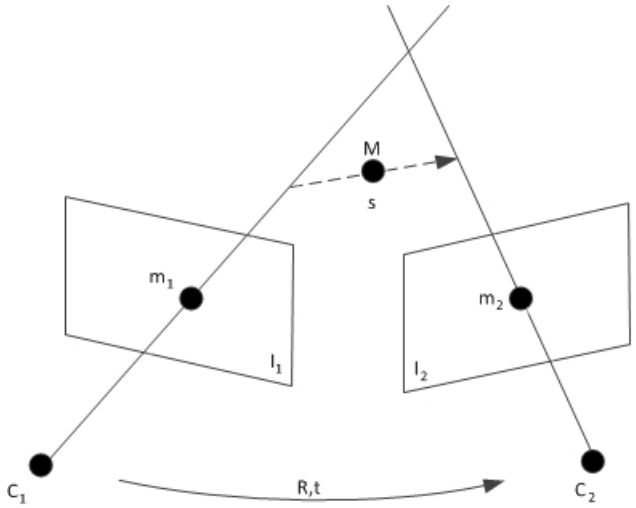


Abbildung 2.5: Rekonstruktion eines 3-D-Punktes mittels Stereotriangulation nach [Sch05]

Hardwaremäßig kann man für ein Stereo-Kamera-System im Prinzip fast jede Art von Kamera verwenden, sei es eine Webcam, Digitalkamera etc. solange sie eine Videoausgabe zur Verfügung stellen kann. Häufig werden handelsübliche Webcams eingesetzt. Fix fertige Systeme wie die Pointgrey Bumblebee 2 Abbildung 2.6 (a) [8] welche jedoch noch über einen Firewire- und keinen USB Anschluss verfügt. Daher wäre diese Kamera für die Drohne weniger geeignet da am Gumstix Summit kein Firewire Anschluss vorhanden ist.

Eine weitere, sogar relativ preiswerte (60 Euro), Lösung wäre die Minoru3D Webcam Abbildung 2.6 (b) [35]. Diese hat den Vorteil, dass sie für beide Kameras nur einen USB Anschluss anstatt üblich zwei verwendet. Auf die beiden Kameras wird im Zeitschlitzverfahren zugegriffen. Dies bedeutet jedoch, dass beide Kameras nicht 100 % synchron sind.

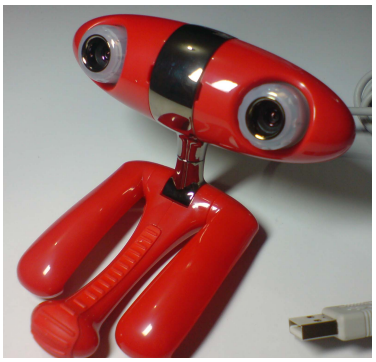
Ein Kamera Modul, das Gumstix CASPA ist bereits vorhanden. Das CASPA Modul wird jedoch nicht über USB sondern über einen ISP Anschluss mit dem Gumstix COM verbunden. Es ist daher nicht möglich eine zweite Kamera vom selben Typ zu verwenden. Für zwei handelsübliche USB Kameras/Webcams müsste ein USB HUB angeschafft werden um den Anschluss der Webcams zu ermöglichen.

2.1.3 Mono-Kamera-Systeme

Bei Mono-Kamera-Systemen treten im Prinzip die gleichen Schwierigkeiten auf wie bei Stereo Systemen. Sie müssen ebenfalls kalibriert werden und die Optik hat mit Verzerrungen zu kämpfen. Ihr großer Nachteil besteht aber darin, dass mit einer reinen Mono Kamera kein direktes tiefenerfassendes Sehen möglich ist. Es benötigt deshalb verschiedene Ansätze um dies trotzdem zu ermöglichen. Einige dieser Möglichkeiten werden nachfolgend diskutiert.

Es gibt aber dennoch einige Möglichkeiten mit einem Mono-Kamera-System Entfernungen zu Objekten im Raum zu bestimmen. Eine Möglichkeit dafür wäre z. B. dass sich die Kamera bewegt. D.h. man hat ein Bild zum Zeitpunkt t_1 an Position 1 und ein Bild zum Zeitpunkt t_2 an Position 2. Durch diese Verschiebung der Positionen der Mono Kamera verhält sich dies ähnlich wie ein Stereo System und man kann z. B. ein Triangulationsverfahren dafür anwenden. Dieser Ansatz

(a) Bumblebee 2 Webcam. Abbildung von Sancho McCann [9] unter CC BY 2.0, abgeändert von Christoph Löffler



(b) Minoru 3D Webcam. Abbildung von Marlon J. Manrique unter CC BY 2.0 [34]

Abbildung 2.6: Stereo-Kamera-Systeme

wird in der Literatur auch Struktur aus Bewegung (structure from motion, SfM) genannt und liefert bereits gute Ergebnisse bei der Rekonstrukton von 3D Szenen [DSTT00] [TK95]. Gegenüber Laser Systemen hat SfM den Vorteil, dass wenn man 3D Szenen rekonstruiert auch direkt die Farbinformationen extrahieren kann. So entsteht nicht nur eine realistischere 3D Ansicht sondern man kann durch die Farbinformationen zusätzliche nützliche Informationen erhalten. Im Falle einer Drohne wäre das z. B. die Bestimmung des Horizonts. Die erhaltenen Daten aus dem SfM Verfahren können dann z. B. mit der PCL Library visualisiert werden [45]. In Abbildung 2.7 sieht man eine Point Cloud die eine Straße mit einem Baum links davon und einem Haus davon rechts zeigt.

Abbildung 2.7: 3D Point Cloud Ansicht, links ist ein Baum zu erkennen, in der Mitte eine Straße und auf der rechten Seite ein Haus. Abbildung von PCL unter CC BY 3.0 [44]

Durch diese Technik kann mit einer nach unten gerichteten Kamera die auf einer Drohne montiert ist ein Landeplatz gefunden werden [JMM05]. Dies wird ohne Zuhilfenahme zusätzlicher Sensoren wie Sonar oder Infrarot Abstandssensoren ermöglicht. Durch die nach unten montierte Mono

Kamera ergeben sich jedoch auch weitere Möglichkeiten die für die ICT Drohne hilfreich wären. Dies wäre z. B. das Schweben an einer fixen Position. Verwendet man nur die typischen, auf einer Drohne zu findenden Sensoren wie Gyrometer, Magnetometer und Acceleromator, also ein 9 DOF System, wird die Drohne über die Zeit gesehen zu driften beginnen und nicht stabil an der vorgesehenen Position verharren [HRHM08]. Dieser Ansatz wäre für die ICT Drohne interessant. Der Gumstix COM könnte mit der CAPSA Kamera sowie einer zusätzlichen USB Kamera ausgestattet sein. Man könnte also eine nach vorne gerichtete und eine nach unten gerichtete Mono Kamera verwenden um einerseits Hindernisse zu erkennen und andererseits ein autonomes Landen sowie Position halten bzw. Schweben zu ermöglichen. Beides sind wichtige Funktionen für eine autonome Drohne.

Optischer Fluss [HS81] kann auch zum Position halten oder Landen eingesetzt werden oder auch auch zur Erkennung von Hindernissen. Durch das vom Optischen Fluss erzeugte Vektorfeld kann jedoch nicht die absolute Entfernung zu einem Objekt gemessen werden. Das Vektorfeld funktioniert nach dem Prinzip, dass nahe Objekte längere Vektoren erzeugen, während weiter entfernte Objekte kürzere Vektoren besitzen [12]. In Abbildung 2.7 sieht man einmal das Vektorfeld für eine nach vorne gerichtete Kamera sowie einmal das einer nach unten gerichteten. Wie man sieht, sind die Vektoren bei einem Objekt das näher zum Flugzeug ist, länger. Zwischen den zwei Vektorfelder besteht jedoch noch ein anderer Unterschied. Bei dem nach vorne gerichteten Vektorfeld ist in der Mitte der sogenannte Focus of Expansion (FOE) eingezeichnet. Dieser gibt die Richtung an in welche sich das Flugzeug bewegt. Der Focus of Expansion existiert jedoch nur bei quasi "Zoombewegungen". Im Vektorfeld das nach unten gerichtet ist, ist dieser nicht vorhanden da nur eine Bewegung nach vorne ausgeführt wird. Weiters ist zu beobachten, dass sich alle Vektoren vom FOE radial entfernen.

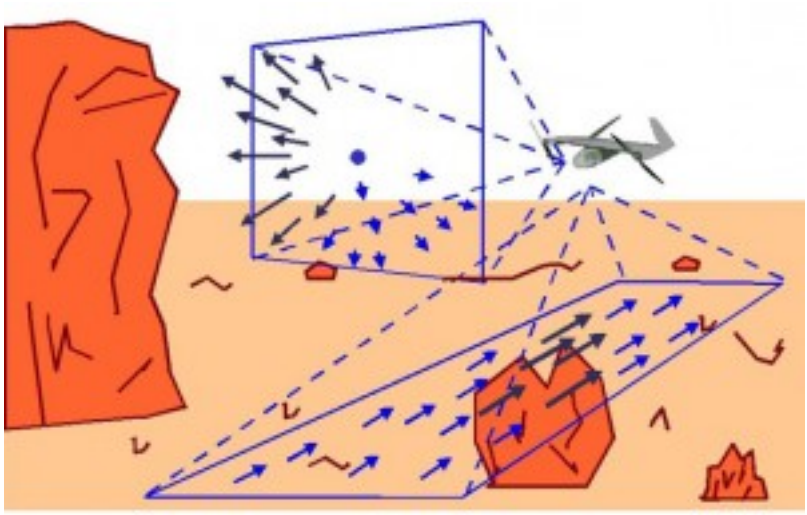


Abbildung 2.8: Optischer Fluss aus Sicht eines Flugzeugs. Abbildung von Centeye, Inc. unter CC-BY [12]

Für eine genauere Abstandsmessung, also um absolute Werte zu erhalten, sind zusätzliche Verfahren notwendig. Time to Contact (TTC) ist eines dieser Verfahren. Dabei wird davon ausgegangen, dass die Geschwindigkeit konstant und vorab bekannt ist. Die Geschwindigkeit kann z. B. durch Odometrie, GPS oder Rotationsencoder ermittelt werden [Cam95]. Hat man die Geschwindigkeit, den FOE und den Optischen Fluss bestimmt, muss die Länge der einzelnen Vektoren des Vektor-

feldes berechnet werden. Der Abstand zum FOE und die Länge der Vektoren gibt dann an wie weit der jeweilige Punkt bzw. das Objekt entfernt ist. In der Tierwelt kann ein solches Verhalten ebenso beobachtet werden [SB12].

Eine weitere Möglichkeit, die auch für Augmented Reality (AR) Anwendungen verwendet werden, sind Marker. Gerade im AR Bereich ist es wichtig, dass die Projektion von 2D Punkten in eine 3D Szene korrekt ist da es sonst vorkommen kann, dass z. B. die Perspektive verzerrt ist oder die Maße nicht stimmen, so das z. B. ein Würfel verzerrt ist. Sind die Maße der Marker vorab bekannt, könnte man sich die Entfernung zum Marker und die Richtung/den Blickwinkel bestimmen. Marker können verschiedene Muster besitzen, in [LSP08] wurde für ein autonomes Landen ein Kreismuster verwendet. Während sich in der Umgebung platzierte Marker durchaus zum Landen eignen oder alternativ auch als Markierung für markante Punkte. Als Beispiel könnte hierfür z. B. ein Paketabwurfpunkt für eine Paketlieferdrohne sein, da die Marker (sofern z. B. der Marker ein QR Code ist) auch Informationen enthalten können.

2.1.4 Laser Systeme

Laser Systeme sind als aktive Systeme ausgeführt. Das bedeutet je nach eingesetztem System wird die Entfernung unterschiedlich ermittelt. Eingesetzte Methoden sind je nach Anwendung meist Triangulation, Laufzeitmessung oder auch die Messung über die Phasenlage.

Laserscanner haben den Vorteil, dass sie selbst über große Distanzen die Entfernung sehr genau messen können. Noch dazu haben sie einen sehr großen FOV (Field of View) Bereich. In [FFH] wird eine Hubschrauberdrohne mit einem Rotordurchmesser von drei Metern und einem Gewicht von knapp 83.5 kg verwendet die mit einem SICK Laser Scanner bestückt ist. Um Gewicht zu sparen, wurde das Gewicht des Laserscanners von ursprünglich 4.5 kg auf 1.6 kg reduziert. Der ICT Hexacopter wäre mit einem Laserscanner von 4.5 kg überfordert und könnte wohl nicht starten. Selbst wenn man das Gewicht auf 1.6 kg reduzieren würde, wäre ein Flug theoretisch möglich aber man müsste große Einschnitte in der Agilität sowie Flugdauer hinnehmen. Außerdem würde sich die effektive zusätzliche Nutzlast auf 400 g reduzieren, da der ICT Hexacopter auf eine maximale Nutzlast von 2 kg ausgelegt ist.

In [MPRPF98] wurde ein SICK PLS 100 Laser Scanner eingesetzt. In Abbildung 2.9 ist ein neuerer Laserscanner der Firma SICK zu sehen. Hier wurde jedoch kein Flugmodell sondern ein Auriga-α Roboter verwendet, der aufgrund des unebenen Geländes mit Ketten wie ein Panzer statt Rädern ausgestattet wurde. Objekte konnten sicher erkannt und es konnte ihnen ausgewichen werden. Hervorzuheben ist der geringe Rechenaufwand, da in der Arbeit nur ein relativ alter Pentium Prozessor verwendet wurde.

In [Kra12] wurde ein Hokuyo UTM-30LX Laser [HAC09] Scanner eingesetzt. Dieser wiegt nur 210 g (ohne Kabel) und hat eine FOV von 270° sowie Reichweite von bis zu 30 m bei +/-50 mm Genauigkeit. Durch das geringe Gewicht, die hohe Genauigkeit sowie den großen FOV würde sich dieser Sensor gut für den Hexacopter eignen. Einzig der Stromverbrauch ist mit bis zu 8 W zwar nicht ganz so hoch wie die bereits genannten SICK Geräte aber dennoch beträchtlich im Vergleich zu z. B. Kamera Systemen. Der sehr hohe Preis von umgerechnet fast 4100 Euro [26] macht das Gerät uninteressant.

Laserscanner müssen jedoch nicht immer schwer sein und viel Energie verbrauchen. Wie in [GMS+08] wurde ein Laserscanner, welcher nur 26 g wiegt, speziell für kleine unbemannte Luftfahrzeuge entwickelt. Ein offizieller Preis konnte nicht in Erfahrung gebracht werden, jedoch ist

Abbildung 2.9: SICK LMS 200. Abbildung von SICK AG

laut [36] mit Kosten in Höhe von knapp 2000 Euro zu rechnen. Ansonsten wäre dieser Sensor aufgrund seiner Eigenschaften sehr gut für die Hinderniserkennung am ICT Hexacopter geeignet. Durch das geringe Gewicht könnte man auch in Betracht ziehen mehrere davon zu montieren um eine komplette 360 ° Rundumsicht zu erhalten. Dies würde für einen Hexacopter, welcher sechs Freiheitsgrade besitzt durchaus Sinn machen. Dadurch wäre auch gleich die Möglichkeit gegeben den Boden zu scannen und so einen geeigneten Landeplatz zu eruieren.

Während die größeren und schweren Laser Scanner vorwiegend auf Fahrzeugen verwendet werden. Entweder zur Evaluierung der mit anderen Systemen gewonnen Ergebnisse wie in [CEBENCP08] oder direkt als Hinderniserkennung wie in [MPRPF98].

Zwei negative Gemeinsamkeiten haben jedoch alle Laser Scanner, einerseits den sehr hohen Preis und andererseits, dass man keine Farbinformationen gewinnen kann. Dies ist zwar für eine reine Hinderniserkennung nicht notwendig, in Bezug auf die Erweiterbarkeit des Einsatzgebietes ist dies jedoch ein großer Kritikpunkt. Dadurch wäre es z. B. nicht möglich 3D Karten mit der Farbinformation, die der realen Welt entspricht, zu erstellen.

2.1.5 Kinect

Die Kinect wurde von Microsoft Ende 2010 [33] vorgestellt. Dabei handelt es sich um ein Hybrides System welches mehrere Sensoren zur Bildverarbeitung in einem Gerät vereint Abbildung 2.10 (a). Als Sensoren sind eine RGB Kamera eine IR LED sowie eine IR Kamera und zusätzlich noch ein Mikrofon und ein drei Achsen Beschleunigungssensor verbaut [32].

Seit der Markteinführung erfreut sie sich großer Beliebtheit [Zha12]. Dies liegt wohl zum einen am relativ geringen Preis von derzeit 100 Euro und zum anderen an den vielfältigen Möglichkeiten die die Kinect ermöglicht. Die Kinect kann z. B. zum Erkennen von Objekten und zum Abstandmessen verwendet werden. Durch das von Microsoft gratis zur Verfügung gestellte Software Development Kit (SDK) ist es sehr einfach möglich auf die Hardware zuzugreifen. Microsoft bietet auch Beispielcodes an, um direkt eine 3D Pointcloud zu erhalten [31].

Um Tiefeninformationen der Szene zu gewinnen verwendet die Kinect strukturiertes Licht. Das Licht wird durch den IR Projektor ausgestrahlt und durch die IR Kamera wieder erfasst,durch einige Berechnungen wird die 3D Szene rekonstruiert wie in Abbildung 2.10 (b) zu sehen ist. Durch die zusätzliche RGB Kamera kann die Szene zusätzlich mit Farbinformationen versehen

werden. Der als sinnvoll erachtete Arbeitsbereich der Kinect wird als ca. 1 - 3 m angesehen. Außerhalb dieses Bereiches wird die Messung zunehmend ungenauer [Kho11]. Der Arbeitsbereich wäre für den ICT Hexacopter durchaus ausreichend. Mehrere Projekte verwenden die Kinect zur Steuerung einer Drohne z. B. [SHBS11a], [SHBS11b] u. [HBH+11].

Wenn man diese Arbeiten vergleicht wird einem auffallen, dass diese Arbeiten eines gemeinsam haben: sie verwenden die Kinect und die Drohne nur in Innenräumen. Dies hat den einfachen Grund, dass bei zu heller Umgebung, z. B. bei Sonnenlicht, keinerlei Tiefeninformationen der Szene gewonnen werden können. Da die ICT Drohne für den Außenbereich konzipiert wurde schließt dieser Umstand den Einsatz des Kinect Sensors aus.

(a) Kinect [30]

(b) Beispiel für eine von der Kinect erzeugte Ansicht der Szene [ZSS+13]

Abbildung 2.10: Kinect

2.2 Akustische Systeme

In den folgenden zwei Unterpunkten werden zwei häufig in der Robotik verwendete Sensoren diskutiert und auf ihre Eignung als Einsatz auf der ICT Drohne überprüft. Bei den Sensoren handelt es sich, anders als bei Mono- oder Stereo Systemen, um aktive Sensoren.

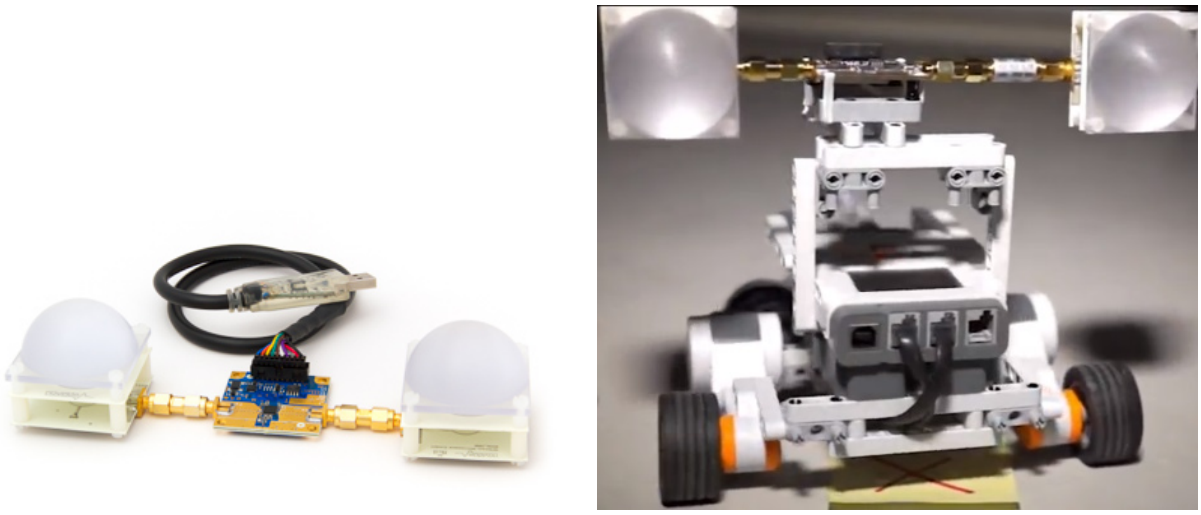
Beide Verfahren funktionieren nach dem Prinzip des Laufzeitverfahrens. Dabei werden bei einem Radio Detection and Ranging (RADAR) System Impulse ausgesendet welche von einem Objekt zurück zum Sender reflektiert werden. Dort kann dann die Zeit bis zum Empfang der Reflektion gemessen und so die Entfernung bestimmt werden. Ebenso wie RADAR funktioniert auch Ultraschall mit dem Unterschied, dass Ultraschall wie der Name schon sagt in einem höheren Frequenzbereich, meist auch mit weniger Leistung als RADAR, arbeitet. Die Frequenzen für Ultraschallsensoren fangen über dem hörbaren Bereich des menschlichen Ohres an, also etwa ab 16 - 20 Khz.

2.2.1 Radar

Radar Systeme wie man sie normalerweise kennt sind für eine Drohne mit den Ausmaßen des ICT Hexacopters nicht geeignet. Einerseits aufgrund der Größe aber auch wegen des Stromverbrauches. Da aber keine Reichweiten der Objekterkennung im Kilometer Bereich notwendig sind, kann auf

kleineren Geräte die in Ultra Breitband (UWB) senden, zurückgegriffen werden. Damit Geräte als UWD bezeichnet werden dürfen, müssen diese mindestens eine Bandbreite von 500 Mhz aufweisen [Ada10]. UWD sendet kurze Impulse aus, diese werden vom Objekt reflektiert und dadurch kann die Entfernung gemessen werden. Um über einen gesamten Operationsbereich des Radars die Entfernung zu messen, sind für jeden Bereich einzelne Impulse notwendig. Dies hat zur Folge, das Messungen länger brauchen. Um dieses Problem zu umgehen kann man statt mit nur einem Sampler die Daten zu verarbeiten, mehrere Sampler gleichzeitig verwenden [NB]. Einer der nachfolgend vorgestellten Chips vom Hersteller Novelda besitzt z. B. 256 Sampling Einheiten.

Als Beispiel für UWD Radar Sensoren sei das Micro Radar Altimeter, also ein Höhenmesser von der Firma smartmicro angeführt [sma12]. Dieser Sensor wurde speziell für den Einsatz in UAVs entwickelt und wiegt in der leichtesten Ausführung nur 160 g und bietet eine Reichweite von bis zu 500 m. Anders als der Name andeutet kann das Radar der Firma smartmicro nicht nur zum Messen der aktuellen Höhe verwendet werden, sondern auch zum Messen von Abständen zu anderen Objekten und auch zum Warnen vor Kollisionen [sma12]. Die Kosten dieses Radar Sensors konnten nicht in Erfahrung gebracht werden. Da smartmicro jedoch den Anschein macht ihre Produkte vorwiegend für den professionellen Markt vorzusehen, wird der Sensor höchstwahrscheinlich nicht im niedrigen dreistelligen Eurobereich zu finden sein. Ein Sensor einer anderen Firma, der MRA Type 2, ist ebenfalls für den Einsatz in Drohnen konzipiert. Die Firma wirbt u.a. damit, dass dieser in den CamCopter Drohnen des österreichischen Drohnenherstellers Schiebel verbaut wird. Mit einem Durchschnittsstromverbrauch von 7 W während den Messungen und bis zu 29 W Spitzen sieht man, dass sich dieser Sensor eher für größere Drohnen, wie eben den CamCopter, eignet [sma13]. Als Sensor für die ICT Drohne würde sich auch das NVA-R661 Development Kit anbieten Abbildung 2.11 (a). Dieser Sensor eignet sich sehr gut für die Robotik und im weiteren Sinne auch für Drohnen. Um diesen Umstand zu verdeutlichen, hat der Hersteller Novelda einen Lego Mindstorm Roboter mit diesem Modul ausgestattet Abbildung 2.11 (b) und so den Radarsensor sozusagen zu dessen Augen werden lassen um Objekte zu erkennen [37]. Ein Preis des Development Kits konnte, wie auch bei den zuvor angesprochen Sensoren, nicht ermittelt werden.



(a) Novelda NVA-R661

(b) Lego Mindstorm Roboter mit Novelda Radar Modul

Abbildung 2.11: Novelda Radar. Abbildung von Novelda AS

Da man mit einem Radar auch durch Wände hindurchsehen kann, würden sich interessante neue Anwendungen daraus ergeben. Man könnte z. B. die Drohne für Rettungseinsätze verwenden.

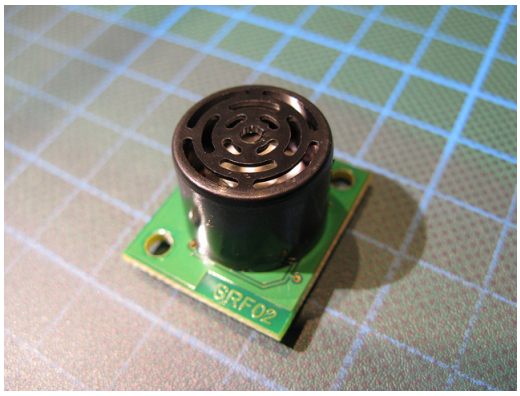
Anders als bei der in Kapitel 1 angeführten Drohne wäre es dann auch möglich z. B. Lawinenverschüttete zu orten, auch wenn sie keinen Lawinenpiepser oder ähnliches am Körper tragen. Oder aber, sollte sich eine Drohne in Innenräumen befinden, könnten so schneller 3D Karten der Umgebung erstellt und dadurch ein optimaler Weg geplant werden.

2.2.2 Ultraschall

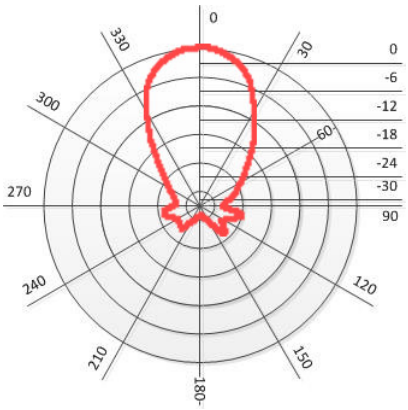
Ultraschallsensoren, haben im Vergleich zu einigen anderen Sensoren den Vorteil, dass diese relativ kompakte Ausmaße sowie ein geringes Gewicht mit sich bringen. Sie sind relativ robust, was ihre Einsatzmöglichkeiten angeht. Das heißt, sie können Entfernungen auch bei schwierigen Bedingungen messen wie z. B. wenn die Oberfläche des Objektes aus Glas besteht oder aber auch bei Nebel. Auch sind ihre Anschaffungskosten sehr überschaubar. Ein SRF02 Ultraschall Entfernungssensor kostet derzeit 14.52 Euro[50] Abbildung 2.12 (a). Dieser Sensor bietet einen Messbereich von 15 cm bis 6 m [Unb]. Der Messbereich wäre für die ICT Drohne ausreichend. Ein Gewicht von 4,6 g, sowie die Anschlussmöglichkeit über I2C sprechen auch dafür. Das Auslesen der Entfernungswerte ist relativ einfach möglich und auch einige Beispielcodes sind im Internet bereits verfügbar. Dies würde eine schnelle Implementierung ermöglichen.

Ein Nachteil dieses Sensors ist jedoch dessen Richtcharakteristik. Wie man in Abbildung 2.12 (b) sehen kann verfügt der SRF02 Sensor über einen relativ schmalen Bereich in dem zuverlässig über größere Distanzen des Messbereichs Objekte erkannt werden können. Die etwa 30 Grad Öffnungswinkel reichen nicht aus. Um eine vollständigere Abdeckung im Sicht- bzw. Flugbereich der Drohne zu erreichen, wären mindestens zwei besser sogar drei Sensoren nötig. Andere Ultraschallsensoren wie der SRF10 bieten einen etwas größeren Öffnungswinkel an, dieser kostet jedoch fast das Dreifache des SRF02.

Bei kommerziellen Drohnen wie der AR.Drone werden ebenfalls Ultraschallsensoren als Unterstützung eingesetzt. Die AR.Drone setzt auf zwei Stück Ultraschallsensoren der Firma Prowave [BCV+11]. In [GMM12] wurde ein Quadrocopter für die Hinderniserkennung mit mehreren, in Summe 12, Ultraschallsensoren bestückt. Dies wurde gemacht um die bereits zuvor angesprochene Problematik mit dem geringen Abstrahlwinkel zu kompensieren und auch um eine 360 Grad Abdeckung zu erhalten. Außerdem wird ein Bereich immer von mehreren Sensoren gemessen um die Zuverlässigkeit zu erhöhen. Gewichtsmäßig würden auch 12 Sensoren für den ICT Quadrocopter kein Problem darstellen, die Kosten würden jedoch erheblich steigen. Ein weiteres Problem das in der Arbeit angesprochen wurde ist, dass die Sensoren sich gegenseitig Stören. Ein solches Problem kann bei passiven Ansätzen eigentlich nicht auftreten. Weiters ist zu beachten, dass es vorkommen kann, dass nicht die gesamte Szene erfasst wird. Man stelle sich eine Wand vor, vor der ein Würfel platziert ist. Verwendet man jetzt nur einen Ultraschallsensor und die Drohne bewegt sich geradewegs auf den Würfel zu, wird man zwar die Entfernung zum Würfel recht genau messen können, jedoch kann es sein, dass die Entfernung zur Wand aufgrund des zu schmalen Kegels des Ultraschalls nicht gemessen wird. Dies hat zur Folge, dass eine Kollision zwar vermieden werden kann, ein Navigieren durch die Umgebung dadurch aber erschwert wird. Durch mehrere, geschickt positionierte Sensoren kann dies jedoch verhindert werden. Der Aufwand die gewonnen Daten von mehreren Ultraschallsensoren zu verarbeiten und aufzubereiten würde dadurch allerdings erheblich ansteigen.



(a) SRF02 Sensor. Abbildung von Lenz Grimmer unter CC BY-NC-SA 2.0 [49]



(b) SRF02 Richtcharakteristik nach [Unb]

Abbildung 2.12: In Abbildung (a) sieht man einen SRF02 Ultraschallsensor, welcher u.a. aufgrund seines Preises gerne in der Robotik als Distanzsensor verwendet wird. In Abbildung (b) ist die Richtcharakteristik des Ultraschallsensors abgebildet. Wie man sieht, besitzt dieser einen Öffnungswinkel von ca. 30 Grad.

2.3 Auswahl des Systems

Wie man anhand der vorigen Unterpunkte sehen kann, gibt es eine Vielzahl an verschiedenen Möglichkeiten um ein System für die Hinderniserkennung zu realisieren. Manche davon sind etwas einfacher zu implementieren oder zielen nur auf den Zweck ab, Entfernungen zu Objekten (und dadurch auch Hindernisse) zu erkennen. Während andere Systeme wie Mono- und Stereo-Kamera-Systeme noch einen deutlichen Mehrwert hinzufügen, da es damit z. B. durch Structure from Motion Technik möglich ist, 3D Karten der Umgebung zu erstellen und diese Karten anders als bei Laser Systemen auch mit Farbinformationen zu versehen um sie realistischer wirken zu lassen.

In der Tabelle 2.1 soll nochmal in sehr verkürzter Form auf die Vor- und Nachteile der Systeme eingegangen werden. Dazu wird ein einfaches Bewertungssystem herangezogen. Ein + steht für gut, o für neutral und ein - für weniger gut. Alle Werte sind relativ zu dem Besten, bzw. zu den anderen vorgestellten Systemen zu sehen. Die Kinect ist zwar ein sehr interessantes System, aber aufgrund der Untauglichkeit im Freien zu operieren, scheidet sie aus und wird nicht in der Tabelle aufgeführt.

	Stereo Kamera	Mono Kamera	Laser System	Radar	Ultraschall[1]
Erweiterbarkeit	+	+	-	o	o
Realisierungsaufwand	+	o	+	+	+
Reichweite	o	o	+	+	o
Gewicht	+	+	o	+	+
Kosten	+	+	-	-	+

Tabelle 2.1: Tabellarischer Vergleich der Systeme

[1] Bezogen auf einen Ultraschall Sensor, für eine sinnvolle Realisierung sind mehrere notwendig

Wie man anhand der Tabelle nochmal sehen kann, haben sowohl Laser- als auch Radar Systeme einen deutlichen Vorteil bei der Reichweite der Hinderniserkennung gegenüber anderen Systemen, aber ihre Kosten sind vergleichsweise sehr viel höher als die der vergleichbaren Systeme.

Ultraschall Sensoren bieten eine gute und genaue Möglichkeit Entfernungen zu messen, ihre Erweiterbarkeit ist jedoch sehr eingeschränkt. Weiters sei zu beachten, dass man meist nicht nur einen Sensor benötigt sondern mehrere um gute Ergebnisse zu erzielen. Mehrere Sensoren wiederum erhöhen die Kosten, Gewicht und Komplexität der Software. Durch diese Faktoren sind Ultraschallsensoren für den Einsatz auf der ICT Drohne weniger gut geeignet.

Stereo und Mono-Kamera-Systeme sind relativ ähnlich zu bewerten. Ein deutlicher Vorteil, bedingt durch die Verwendung von zwei Kameras, von Stereo Systemen gegenüber Mono Systemen ist jedoch, dass sie ohne Bewegung die absolute Entfernung zu Objekten bestimmen können. Darum würde die Wahl für die Realisierung eher auf ein Stereo-Kamera-System fallen. Da aber bereits ein Gumstix Caspa Kamera Modul vorhanden ist, wodurch ohne große Leistungseinbußen des Systems hinnehmen zu müssen Kosten eingespart werden können und auch ein absolutes Entfernungsbestimmen möglich ist sobald die Drohne sich bewegt, fällt die Wahl des zu realisierenden Systems auf ein Mono-Kamera-System basierend auf dem Caspa Kamera Modul.

3 Hinderniserkennung mittels optischen Fluss

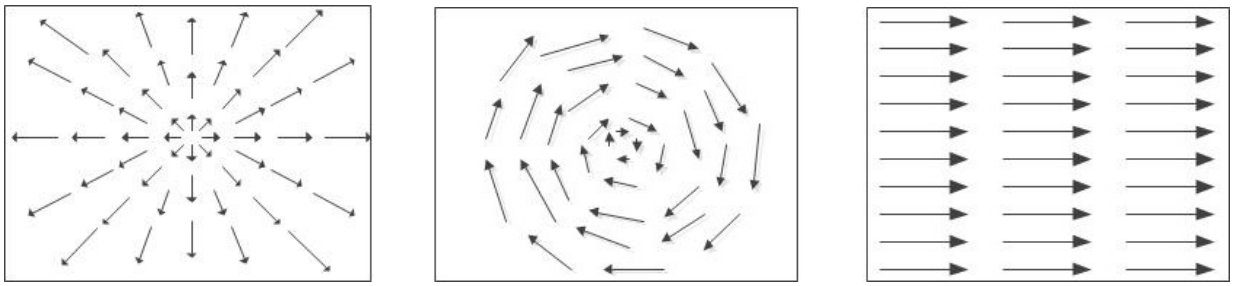
In diesem Kapitel wird auf die Implementierung der Hinderniserkennung mittels Mono Kamera System, dessen Vorzüge in Kapitel 2 ausreichend diskutiert wurden, eingegangen. Dabei wird kurz zwei Optische Fluss Algorithmen eingegangen. Anschließend werden Probleme bei der Umsetzung der Diplomarbeit auf der aktuellen Drohnen Plattform diskutiert und Verbesserungsmöglichkeiten sowie Lösungsansätze diskutiert. Für die neue Plattform wurde ein Evaluierungsprogramm geschrieben welches die Performance und Eignung aufzeigen soll. Die Ergebnisse der Evaluierung werden in Kapitel 4 besprochen. Außerdem beinhaltet dieses Kapitel noch die Details der Implementierung der Hinderniserkennung wie z.B. die Feature Detektion, Berechnung des Fokus of Expansion und weitere wichtige Punkte welche für die Umsetzung notwendig sind.

3.1 Optischer Fluss

Als Hauptalgorithmus wird der Optische Fluss eingesetzt, welcher in diesem Kapitel näher erläutert wird. Es werden nicht nur die zwei großen Vertreter von Algorithmen angesprochen, sondern auch die Umsetzung dieser hinsichtlich der Hinderniserkennung der Drohne.

Der Optische Fluss gibt ein Vektorfeld zurück, das die Bewegung von Objekten, der Kamera oder auch eine Kombination aus beidem widerspiegelt. Dabei geben die Vektoren, die mittels eines Optischen-Fluss-Algorithmus berechnet werden, die Richtung sowie Geschwindigkeit für einen Bildpunkt wider. Die Vektorfelder haben je nach Art der Bewegung eine unterschiedliche Form. Die drei Grundbewegungen werden in Abbildung 3.1 (a-c) dargestellt. Da ein Multikopter (also auch die ICT Drohne) jedoch 6DOF (6 Degree of Freedom) also sechs Freiheitsgrade besitzt und ein komplett ruhiger Flug vorwärts entweder durch äußere Einwirkungen (z. B. leichter Wind) oder aber einfach nur aus dem Grund, dass man diverse Flugmanöver durchführen will, unmöglich ist, ergibt sich immer ein Vektorfeld, welches sich aus den verschiedenen Grundbewegungen zusammensetzt. Auch bei einer Vorwärtsbewegung hat man durch die Art des Antriebes außer bei ganz langsamen Flug eine leichte Veränderung (Drehung) um die X Achse, in der Flugtechnik auch Tilt genannt.

Beim Optischen Fluss wird zwischen zwei großen Gruppen unterschieden:dichte (engl. dense) Verfahren wie z. B. das von Farnebäck [Far03] oder spärliche (engl. sparse) Verfahren wie das von Horn und Schunck [HS81] oder Lucas und Kanade [LK+81]. Auf diese beiden wichtigen Verfahren wird in den nächsten Unterpunkten noch im Detail eingegangen. Das Verfahren von Lucas



(a) Vorwärts Bewegung

(b) Rotation

(c) Translation nach rechts - die Vektoren bewegen sich immer in die entgegengesetzte Richtung

Abbildung 3.1: Vektorfeld der drei Grundbewegungen des Optischen Flusses, Abbildungen nach [MBMG01]

und Kanade, ist einer der am häufigsten verwendeten Algorithmen, wenn es um Fast-Echtzeit- oder sogar Echtzeit-Berechnung des optischen Flusses geht. Warum dies so ist, ist einfach erklärt. Während dense optical flow Algorithmen versuchen für jeden Pixel einen Vektor zu errechnen, wird bei sparse optical flow Algorithmen nur die Bewegung einer Summe aus vorab ausgewählten Pixeln berechnet. Geht man nun davon aus, dass ein Bild eine Größe von 640x480 Pixeln (diese Auflösung kann das CASPA Camera Modul bereitstellen) hat, würde dies 307200 Pixeln entsprechen. Da die Berechnung von 307200 Pixeln um einiges mehr an Zeit in Anspruch nimmt, als eine Berechnung von angenommen nur 100-500 Pixeln (im Fall dieser Arbeit handelt es sich um Pixel, welche markante Bildpunkte darstellen) und die Hardwareressourcen begrenzt sind ist es natürlich von Vorteil weniger Bildpunkte berechnen zu müssen.

Algorithmen zum Optischen Fluss sind bereits schon seit vielen Jahren bekannt, aber sie sind immer noch für die Forschung interessant und es wird immer noch nach Möglichkeiten und Ansätze wie man die Algorithmen verbessern kann gesucht. Horn und Schunck verfassten eine Arbeit, welche 1981 veröffentlicht wurde [HS81], in der sie eine Möglichkeit der Berechnung für den Optischen Fluss beschreiben. Bereits in den 50er Jahren wurde die visuelle Wahrnehmung des Optischen Flusses von Gibson [Gib50] beschrieben. Ein großer Nachteil bei der Berechnung des Optischen Fluss ist, dass diese mit relativ viel Aufwand verbunden ist. Dadurch konnten lange Zeit keine Anwendungen verwirklicht werden, bei welchen eine Berechnung in Echtzeit notwendig ist. Mittlerweile hat sich dieser Umstand u.a. durch den Einsatz von sparse optical flow Methoden, wie etwa durch jene von Lucas und Kanade [LK+81] verbessert. Auch die Weiterentwicklung der Prozessoren welche heutzutage eine viel höhere Leistung erzielen als noch vor beispielsweise zehn Jahren, hat dazu beigetragen, dass der Optische Fluss mittlerweile auch in Echtzeitanwendungen eingesetzt werden kann. Auch das Auslagern der Berechnungen auf eine Grafikkarte (GPU) oder auf Digitale Signalprozessoren (DSP) hat dazu beigetragen. Grafikkarten in Standard PC Bauweise sind durch ihren hohen Leistungsverbrauch sowie Größe eher weniger für einen Einsatz auf einer Drohne geeignet. Der bereits auf der Drohne verbaute Gumstix COM, welcher einen OMAP 3530 Chip inkl. C64x+ DSP [Ins13] verbaut hat sollte sich für die Anforderung besser eignen. Der geringe Leistungsverbrauch sowie Gewicht und Größe sprechen eindeutig dafür.

Einen Überblick über die Laufzeiten derzeit gängiger Verfahren bietet die KITTI Vision Benchmark Suite der Technischen Universität Karlsruhe, welche nicht nur für den Optischen Fluss sondern auch für Stereo oder Odometrie Vergleichsergebnisse der Laufzeit der einzelnen Algorith-

men und Datasets zur Verfügung stellt [29] [GLSU13].

Der Optische Fluss kann auch in der Tierwelt, [SZLC96] vor allem bei Insekten, beobachtet werden und dient dort, genauso wie es in dieser Arbeit der Fall sein soll, zum Erkennen und Ausweichen von Hindernissen oder auch zum Anpassen der Geschwindigkeit um sicher durch enge Passagen zu navigieren. Das Schema, das in der Tierwelt beobachtet werden konnte ist heutzutage ein verbreitetes Thema in der Forschung wenn es um Visuelle Navigation oder auch um Odometrie geht [SCW+99].

3.1.1 Horn und Schunck

Der Algorithmus von Horn und Schunck [HS81] war einer der ersten zur Berechnung des Optischen Flusses. Es handelt sich hierbei um ein globales Verfahren, bei welchem der Optische Fluss über das gesamte Bild berechnet wird. Bei der Berechnung des Verfahrens von Horn und Schunck werden einige Annahmen getroffen um die Berechnung zu ermöglichen. Hier wird nur kurz auf die Annahmen und das allgemeine Prinzip hinter der Methode von Horn und Schunck eingegangen. Für die genaueren mathematischen Hintergründe sei auf das Originaldokument [HS81] oder auf [SS99] verwiesen.

Um den Optischen Fluss zu berechnen, gehen Horn und Schunck von zwei zentralen Annahmen aus. Die erste Annahme ist die Optical Flow Constraint Equation oder auch image brightness constancy equation genannt. Dabei wird davon ausgegangen, dass die Helligkeit eines Objektes sich von einem Bild zum nächsten Bild nicht verändert. Mit dieser Bedingung alleine kann jedoch noch nicht die Lösung gefunden werden. Wie in Abbildung 3.2 gezeigt wird, kann das Ergebnis nur dahingehend eingeschränkt werden, dass die Lösung entlang der Linie sein muss. Aus diesem Grund wird eine weitere Bedingung, die Smoothness Constraint, festgelegt.

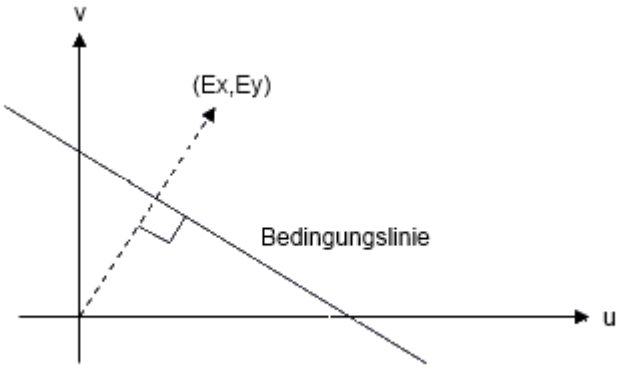


Abbildung 3.2: Brightness Constraint. Hierbei entspricht u und v dem Geschwindigkeitsvektor des Optischen Flusses und Ex und Ey dem Brightness gradient Vektor. Die Bedingungslinie gibt an, wo sich u und v befinden könnten. Erst durch eine weitere Bedingung lässt sich sagen, wo genau sich der gesuchte Punkt auf der Bedingungslinie befindet. Abbildung nach [HS81]

Bei der Smoothness Constraint (Glattheitsbedingung) wird davon ausgegangen, dass sich benachbarte Objekte bzw. Punkte in etwa gleich bewegen. Dies kann man sich in etwa so vorstellen:

bewegt sich ein Objekt im Bild von links nach rechts, weisen alle zu verfolgenden Punkte die sich auf dem Objekt befinden, die selben Bewegungsvektoren (Optische Fluss Vektoren) wie das Objekt selbst auf. Ohne zu sehr ins Detail zu gehen kann man das Verfahren von Horn und Schunck kurz wie folgt beschreiben: Horn und Schunck stellen eine Kostenfunktion auf, welche einen Fehlerwert liefert. Durch iterative Anwendung einer Minimierungsfunktion soll dieser Fehlerwert minimiert werden um so die Komponenten des Optischen Flusses (u,v) genauer bestimmen zu können und nicht nur einen Bereich (Bedingungslinie) wie in Abbildung 3.2 zu erhalten.

3.1.2 Lucas und Kanade

Das Verfahren von Lucas und Kanade [LK+81] wurde genau wie das von Horn und Schunck im Jahr 1981 veröffentlicht. Während der Algorithmus von Horn und Schunck in der Praxis kaum mehr Anwendung findet, ist der von Lucas und Kanade in sehr vielen größeren Bildverarbeitungs-Libraries wie z. B. OpenCV, FastCV oder auch in der Vision Library VLIB von Texas Instruments enthalten. Dieser Aspekt unterstreicht die Bedeutung dieses Algorithmus für die Anwendung in der Praxis auf.

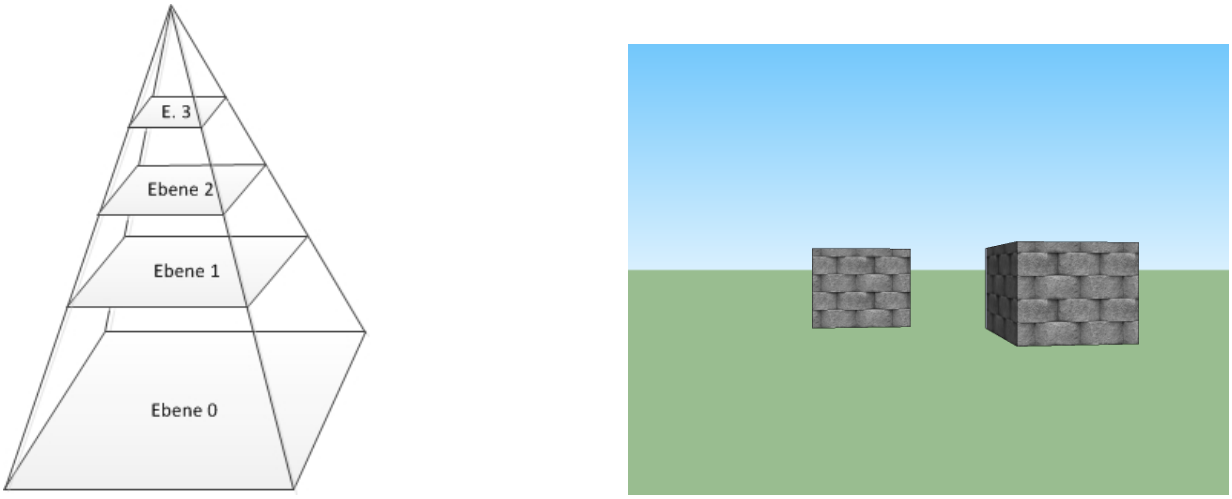
Bei der Verwendung von Lucas und Kanade wird eine Fenstergröße gewählt, welche den Suchbereich um den von einem zuvor z. B. via Feature Detektor ausgewählten Punkt vorgibt. Diese Fenstergröße wird aus Performancegründen meist relativ klein gehalten (wenige Pixel). Das Detektieren größerer Bewegungen ist hier nicht möglich, da sich diese außerhalb des Fensters befinden würden. Dieser Umstand stellt jedoch durch die, im nächsten Unterpunkt beschriebene, mögliche Verwendung von Bildpyramiden kein Problem dar.

Die Bedingung der brightness constancy trifft auf den Algorithmus von Lucas und Kanade ebenfalls zu. Zusätzlich werden noch zwei weitere Bedingungen gestellt. Einerseits wird angenommen, dass ein Objekt nur kleine Bewegungen vollführt und andererseits ähnlich der smoothness constraint, dass Punkte die nahe beieinander liegen, also Nachbarn, eine ähnliche Bewegung vollziehen. Vor allem die zweite Bedingung, dass nur eine kleine Bewegung stattfindet, muss wieder im Zuge dieser Arbeit betrachtet werden. Welche Auswirkung dies hat kann kurz erklärt werden: geht man von der Annahme aus, dass ein Bildverarbeitungsalgorithmus nur fünf Bilder pro Sekunde berechnen kann, würde das bedeuten, dass man das Objekt nur alle 200 Millisekunden sieht. Bewegt man sich nun mit einer höheren Geschwindigkeit, ist der Weg den man innerhalb von 200 Millisekunden zurück legt natürlich auch größer. Aus diesem Grund sollte ein möglicher Algorithmus bzw. die Plattform auf welcher dieser Algorithmus aufgeführt wird eine gute Performance liefern, denn umso mehr Bilder pro Sekunde berechnet werden, umso kleiner ist auch die Bewegung. Dieser Umstand kommt der Berechnung zu Gute.

Ähnlich dem Horn und Schunck Verfahren wäre es ohne den Zusatzbedingungen nicht möglich sich die Vektoren zu errechnen. Im Falle von Lucas und Kanade hätte man zwei unbekannte Variablen (u,v) aber nur eine Gleichung und somit wäre es nicht möglich eine Lösung zu berechnen. Aus diesem Grund verwendet man zusätzlich die benachbarten Pixel um so mehr Gleichungen aufstellen zu können und dadurch je nach Fenstergröße mehr Gleichungen als Unbekannte und somit auch eine Lösung zu erhalten. Dabei verwenden Lucas und Kanade den Ansatz der Methode der kleinsten Quadrate. Für die genaueren mathematischen Hintergründe die in dieser Arbeit nicht im Speziellen behandelt werden, sei genau wie beim Algorithmus von Horn und Schunck auf das Originaldokument oder auf [PU13] verwiesen.

3.1.3 Bildpyramiden

Eine der Annahmen des Lucas und Kanade Algorithmus ist es, dass die Bewegung zwischen den Bildern t_n und t_{n+1} klein ist. Sollte dies nicht der Fall sein, entstehen bei der Berechnung Fehler welche man natürlich vermeiden will. Hierfür werden Bildpyramiden verwendet. Mit ihnen ist es möglich, auch größere Bewegung zu erkennen und dadurch den Fehler in der Berechnung zu vermeiden. Bildpyramiden funktionieren nach einem einfachen Prinzip. Man verwendet das Ausgangsbild mit einer Größe von 640x480 Pixel. Bei jeder neuen Ebene der Pyramide wird nun die Auflösung reduziert, also in der 1. Ebene hat das Bild nur noch eine Auflösung von 320x240, in der 2. Ebene 160x120 und so weiter. Durch das Reduzieren der Auflösung ist nun möglich, auch größere Bewegungen zu berechnen. Man fängt von der höchsten Ebene an und führt dann auf jeder Ebene die Berechnung des Optischen Flusses durch, bis man wieder beim Ausgangsbild der Ebene 0 angelangt ist.Diese Vorgehensweise wurde für den Lucas und Kanade Algorithmus z. B. von Bouguet [Bou01] umgesetzt. In Abbildung 3.3 a-b und 3.4 a-c wird dies genauer veranschaulicht.



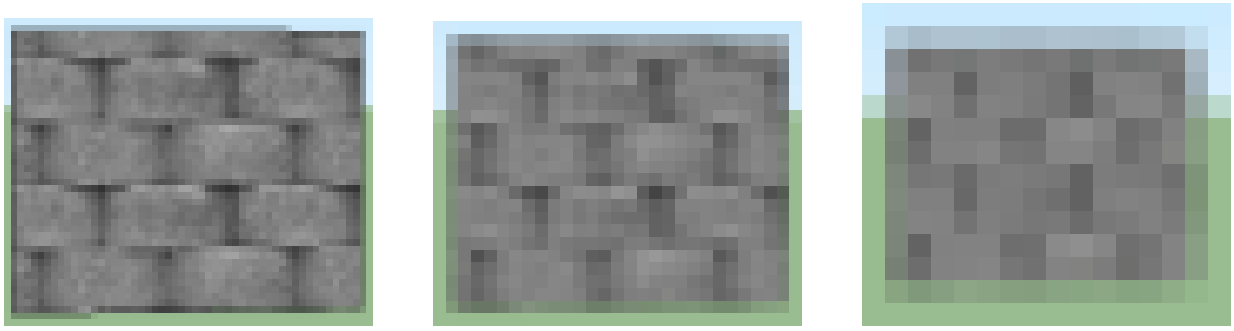
(a) Bildpyramide

(b) Originalbild - Ebene 0

Abbildung 3.3: Bildpyramide sowie die Originalebene, Ebene 0

3.1.4 Schwierigkeiten des Optischen Flusses

Es gibt zwei größere Schwierigkeiten bei der Berechnung des Optischen Flusses. Einerseits das Blendenproblem (Aparture Problem) und andererseits das Korrespondenzproblem [Bra08] [BL10]. Zusätzlich gibt es noch kleinere Optische Illusionen wie die Barbar Pole Illusion, welche die Bestimmung des Optischen Flusses nicht gerade vereinfachen. Das Blendenproblem besteht darin, dass nur ein Teil eines Bildausschnittes sichtbar ist und man dadurch die Bewegung nicht direkt zuordnen kann. In Abbildung 3.6 wird dies anschaulich dargestellt. Dabei sieht man, dass das graue Rechteck sich von oben links nach unten rechts bewegt. Da aber nur ein Ausschnitt des Bildes sichtbar ist, kann die eigentliche Bewegung nicht bestimmt werden und man sieht nur eine reine Bewegung von links nach rechts anstatt der korrekten Bewegung. Aus diesem Grund ist es wichtig, die Fenstergröße (window size) für den Optischen Fluss Algorithmus passend zu wählen. Ist die Fenstergröße zu klein, tritt das Blendenproblem auf, ist sie zu groß wird einerseits die Berechnung langsamer weil ein größerer Ausschnitt betrachtet werden muss, und andererseits die Bedingung verletzt, dass die Bewegung nicht zu groß sein soll.



(a) Ebene 1 - 320x240 (b) Ebene 2 - 160x120 (c) Ebene 3 - 80x60

Abbildung 3.4: Die einzelnen Ebenen der Bildpyramide bis zur 3. Ebene. Dabei wird jeweils der linke Würfel als Bildausschnitt gewählt und dargestellt um die Pixelgröße der entsprechenden Ebenen zu verdeutlichen. Durch den Einsatz von Pyramiden werden die Pixel sozusagen größer und der Berechnungsaufwand sinkt dadurch drastisch. Während die Ebene 0 noch 307200 Pixel hat, hat die 3. Ebene nur noch 4800 Pixel.

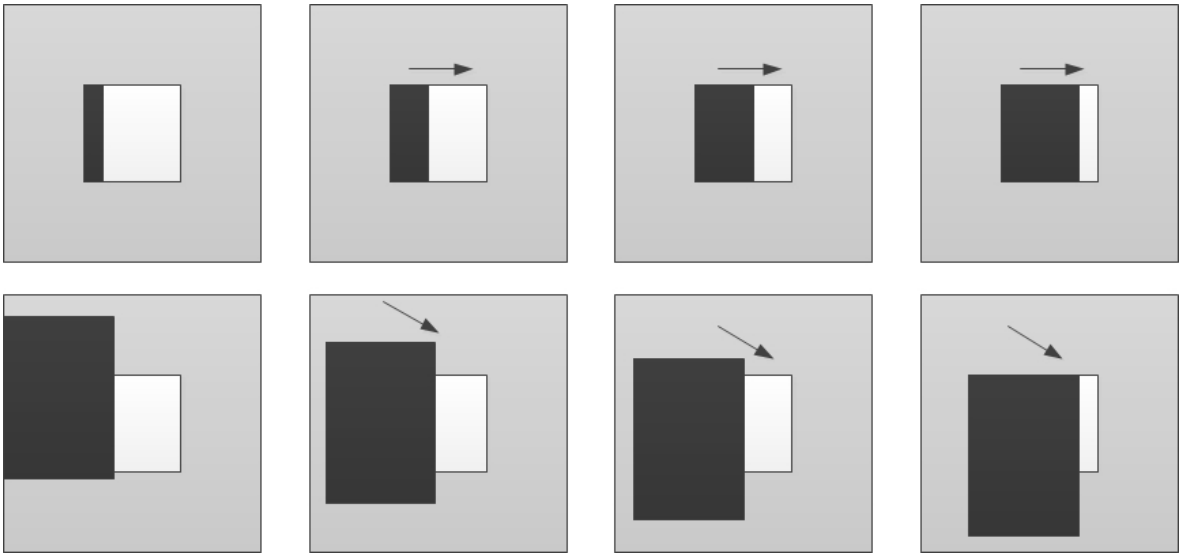


Abbildung 3.5: Blendenproblem - obere Reihe die erkannte Bewegung, untere Reihe die echte Bewegung nach [Bra08]

Das Korrespondenzproblem besteht darin, dass es möglich ist, dass sich Punkte in Bild t_n nicht denen in Bild t_{n+1} zuordnen lassen. Das kann vorkommen wenn die Bilder zu wenig Textur haben wie z. B. ein zu gleichmäßiges Muster (Gitter oder Netz).

Die Barbar Pole Illusion ist ein weit verbreitetes Beispiel für eine Optische Illusion. Diese Illusion führt dazu, dass die Berechnung des Optischen Flusses von der eigentlichen Bewegung abweicht. Normaler weiße geht man davon aus, dass die Berechnung des Optischen Flusses soweit als Möglich dem Bewegungsfeld gleicht, bei der Barbar Pole Illusion ist dies wie in Abbildung 3.7 gezeigt wird nicht der Fall. Da jedoch solche Optischen Illusionen in der Realität in dieser Form eher selten vorkommen, sollte dies für den Betrieb der Drohne vernachlässigbar sein.

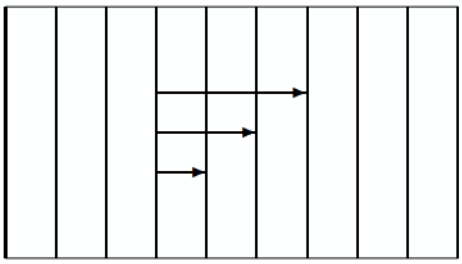
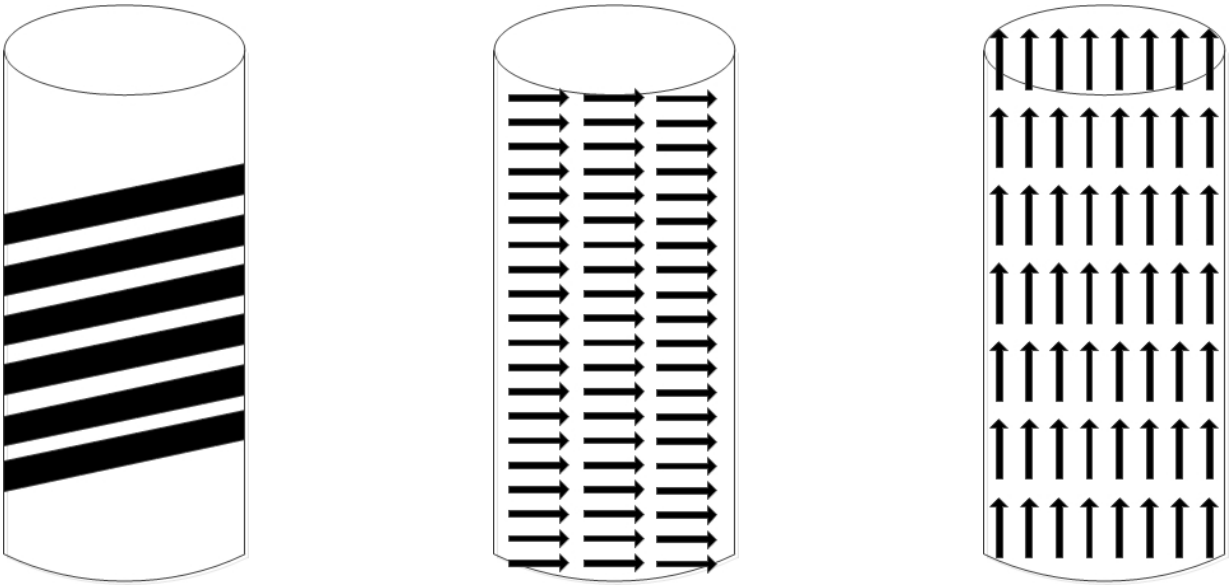


Abbildung 3.6: Korrespondenzproblem - Zuordnung der Bewegung ist nicht möglich



(a) Barbar Pole (b) Bewegungsfeld (c) Optischer Fluss

Abbildung 3.7: Barbar Pole Illusion mit dem Bewegungsfeld und dem Optischen Fluss. Die Barbar Pole dreht sich um ihre eigene Achse. Für das Bewegungsfeld sieht es so aus, als wäre dies eine Bewegung nach rechts. Für den Optischen Fluss hingegen, verursacht durch die diagonal nach oben gehenden Linien, sieht es aus, als würde eine Bewegung nach oben stattfinden. Abbildung nach [6]

3.2 Gumstix Plattform

Als Hauptcomputer befindet sich ein Gumstix Overo Fire COM auf der Drohne, welcher auf einem Summit Extension Board [25] aufgesteckt ist. Dies hat den Grund, dass der Overo Fire keine I/O Ports besitzt (wenn man von dem Anschluss für das Extension Board absieht) und diese nur durch entsprechende Extension Boards nach außen geführt werden können. Die Anschlüsse nach außen sind natürlich essentiell. Einerseits alleine schon wegen eines Anschlusses für die Spannungsversorgung, aber auch um die Verbindung zu weiterer Peripherie herzustellen. In der derzeitigen Ausführung der Drohne werden die externen I/Os nur zum Anschluss an den ATmega Microprozessor verwendet, der über SPI angeschlossen wurde. Der ATmega Mikrocontroller wird eingesetzt um die Lageregelung der Drohne zu verwirklichen um die Periodenzeit von zwei Millisekunden exakt einzuhalten,da der Overo Fire durch Interrupts vom Betriebssystem dazu

nicht gut geeignet ist. Der Mikrocontroller eignet sich dagegen für solche Anwendungen durch die Verwendung von Timern gut. Deswegen wird die Regelung auf dem Mikrocontroller mit einem PID Regler realisiert.

Die Daten die der ATMega Mikrocontroller von den angeschlossenen Sensoren,wie z. B. dem Gyroskop erhält, werden zuerst an den Overo Fire weitergeleitet und von dort anschließend über WLAN an die Kontrollsoftware bzw. Bodenstation, die Peter Hanger [PH12, S. 90] erstellt hat, weitergegeben und visualisiert. Die Software kann jedoch nicht nur Daten visualisieren sondern dient auch dazu, die einzelnen Flugmodi zu testen. Außerdem ist es möglich auch Steuerungsbefehle in umgekehrter Reihenfolge zu senden, also vom PC zum Overo Fire und dann zum Mikroprozessor. Dies ist vor allem sinnvoll, um vor dem Flug z. B. einzelne Motoren auf ihre Funktion zu testen.

Sollte die Drohne in weiterer Folge mit mehr Hardware wie z.B. einem GPS Sensor oder einem UMTS Modem ausgestattet werden, könnte man die Software erweitern und so neue GPS Standortdaten zur Drohe übermitteln und diese automatisch anfliegen lassen. Durch ein UMTS Modem wäre das dann auch über große Distanzen ohne weiteres möglich. Auch der Anschluss solcher Peripherie ist durch das Summit Extension Board, Abbildung 3.8, welches über einen 40 Pin Header, sowie über einen USB Host Anschluss verfügt, problemlos möglich.

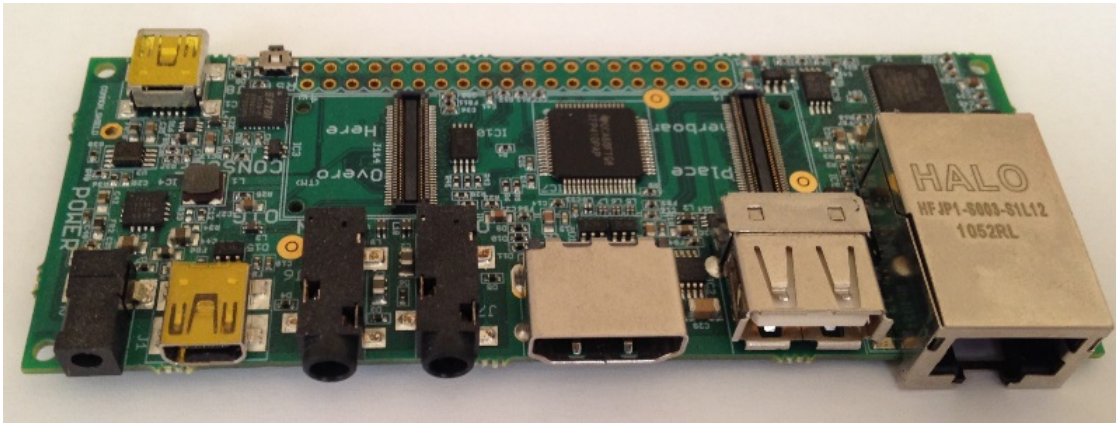


Abbildung 3.8: Summit Extension Board

Der Gumstix Overo Fire bietet auch die Möglichkeit eine Camera, nämlich das CASPA Camera Modul direkt über einen 27 PIN ISP Stecker zu verbinden. Die Idee ist an sich gut da dieser Anschluss relativ wenig Platz am Board verbraucht, jedoch bringt dieser Anschluss auch Nachteile mit sich. Das mitgelieferte Kabel ist mit acht Zentimetern recht kurz geraten. Probleme ergeben sich durch den zentral auf der Oberseite der Drohne montierten Overo Fire, da das CASPA Camera Modul auf der Drohnenunterseite entweder direkt nach vorne oder mit einer Neigung von maximal 45 Grad angebracht werden sollte. Dies ist bei Drohnen die Fotos oder Filmaufnahmen machen üblich, da die Ausleger und Motoren so nicht in der Aufnahme stören. Mit dem acht Zentimeter Kabel gestaltet sich dieses Vorhaben deutlich schwieriger. Gumstix selbst bietet nur diese Kabellänge an. Es gibt zwar längere Kabel von anderen Herstellern, bei denen jedoch erst getestet werden müsste, ob die Datenübertragung über größere Distanzen stabil genug ist. Weiters konnte das Kabel bei der Handhabung nicht überzeugen. Die Stecker sind filigran und halten nicht sonderlich gut. Es wäre durchaus vorstellbar, dass sich das Kabel z. B. bei einer härteren Landung der Drohne etwas löst wodurch natürlich in weiterer Folge die Funktion eingeschränkt wäre. Ein

normaler USB Stecker mit einem Kabel in passender Länge wäre vorzuziehen. Die Vermutung liegt nahe, dass darauf jedoch aufgrund der geringen Ausmaße des Gumstix Overo Fire verzichtet werden musste, da ein USB Anschluss, selbst in Micro Bauweise keinen Platz gefunden hätte. Der Overo Fire besitzt außerdem noch einen MicroSD-Kartensteckplatz auf dem ein kompatibles Betriebssystem installiert werden kann. Dies wird im nächsten Unterpunkt genauer diskutiert.

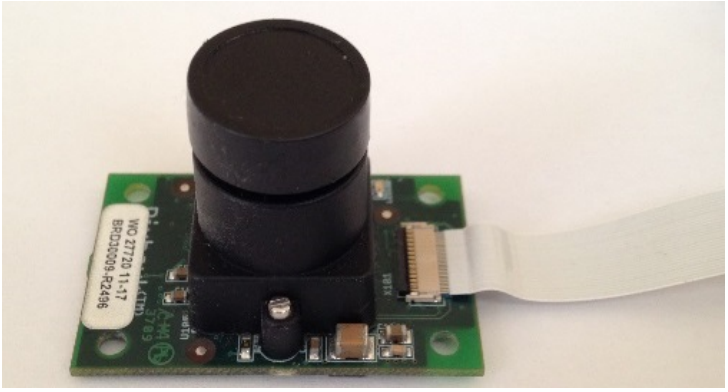


Abbildung 3.9: Gumstix CASPA Camera Modul inkl. Kabel

3.2.1 Overo Fire - Linux Betriebssystem

Der Gumstix Overo Fire ist bereits 2009 auf den Markt gekommen, also mittlerweile fünf Jahre alt. Zu beginn wurde noch Ångström, ein auf Linux basierendes System von Gumstix eingesetzt und unterstützt. In der Zwischenzeit wurde jedoch auf andere Systeme gewechselt. Alternativ zu Angstrom wird auch noch Linaro, ein auf Ubuntu basierendes System sowie das Yocto Projekt angeboten [22]. Das Yocto Projekt Build System ist mittlerweile der Standard für die Gumstix Boards und auch die vorerstellten reinen Konsolen Images oder die Images mit Grafischer Oberfläche basieren darauf. Angstrom wird mit einem alten 2.6.39 Kernel ausgeliefert, während Linaro und die Images basierend auf dem Yocto Build System bereits einen aktuelleren Kernel der 3.5 Version enthalten. Das mag vielleicht auf den ersten Blick nicht unbedingt relevant erscheinen wenn man z. B. nur daran denkt, dass die Drohne nicht direkt mit dem Internet verbunden ist und so gesehen zumindest keine Sicherheitsrisikos durch nicht aktuelle Systeme entstehen. Auf den zweiten Blick wird aber deutlicher, dass ein neuer Kernel viele Vorteile, wie z. B. einen meist besseren Treiber Support, eine bessere Performance und Updates mit sich bringt. Auch hinsichtlich zukünftiger Erweiterungen spricht dies natürlich für den Einsatz eines neuen Kernels bzw. Betriebssystems, das ständig weiterentwickelt wird. Aus diesem Grund wurde auf das neue Yocto basierende System gesetzt. Um gar keine Verwirrung entstehen zu lassen, sei hier angemerkt, dass das Yocto Build System, wie der Name bereits vermuten lässt, keine eigenständige Linux Distribution sondern ein Build System ist. Das mit dem Yocto erstellte Linux System basiert auf OpenEmbedded, genauso wie Angstrom. Zwei Gründe sprechen gegen das Verwenden von Linaro: einerseits ist das System noch relativ neu und gerade im Bezug auf die Gumstix Plattform noch nicht sehr verbreitet. Anderseits verwendet Gumstix selbst Yocto für ihre Images, wodurch mehr Support in Form der Gumstix Mailinglist [23] o.ä. als bei Linaro erwartet werden kann.

Durch diesen Umstand bleibt nun nur noch die Wahl zwischen den bereits zuvor kurz erwähnten vorgefertigten Images oder dem Erstellen eines eigenen Images mit Hilfe des Yocto Build Systems. Als Desktop Image bezeichnet Gumstix das Image mit grafischer Oberfläche. Die Vor- und

Nachteile werden kurz in tabellarischer Form in Tabelle 3.1 wiedergegeben. Wie in der Tabelle zu Ende des zweiten Kapitels wird die selbe Vorgehensweise der Bewertung verwendet. Ein + steht für gut, o für neutral und ein - für weniger gut. Alle Werte sind relativ zum Besten, bzw. zu den anderen vorgestellten Systemen zu sehen.

	Konsolen Image	Desktop Image	selbst erstelltes Image
Download Größe	+	o	-
Installationsaufwand	+	+	-
Installionsdauer	+	+	-
Individualisierung	-	-	+
Ressourcenverbrauch in Betrieb	+	-	+-

Tabelle 3.1: Vergleich der Linux Systeme

Während die Größe des Downloads bei den derzeitig verfügbaren Geschwindigkeiten der Internetanbindungen eine kleine Rolle spielt, sind die anderen Punkte doch relativ wichtig und dürfen auf keinen Fall unterschätzt werden. Wie man anhand der Tabelle deutlich ablesen kann, spricht im Prinzip wenig für den Einsatz eines selbst erstelltes Image und sehr viel für ein Konsolen oder zumindest Desktop Image. Ein selbst erstelltes Image ist jedoch unbedingt nötig da das Konsolen sowie Desktop Image keine DSP Unterstützung mit sich bringen. Dies wird im Detail im Kapitel 3.2.3 OMAP3530 DSP noch genauer erläutert. In dem Unterpunkt wird auch näher ausgeführt, warum man die bereits angesprochen Punkte nicht unterschätzen sollte.

3.2.2 Bitbake

Mit bitbake werden sozusagen einzelne, sogenannte Rezepte (recipes) verarbeitet, um daraus dann ein fertiges Linux Betriebssystem zu erstellen. Bitbake kann in Kombination mit Yocto verwendet werden. Ein Betriebssystem besteht also aus sehr vielen einzelnen Rezepten, von welchen jedes für z. B. ein eigenes Programm oder Modul steht. Der Ablauf [Tea] eines bitbake Prozesses wird in verschiedene Schritte unterteilt:

- Fetch: es werden alle nötigen Quelldateien, meist von einem Repository z. B. GIT oder SVN heruntergeladen
- unpack: Daten werden entpackt
- patch: für die einzelnen Programme werden Patches eingespielt
- configure: konfiguriert die Packages
- compile: aus dem Quellcode wird lauffähiger Programmcode erstellt
- staging: im staging werden Header, Libraries und Programme die von einem Rezept erstellt worden sind auch für andere Rezepte zur Verfügung gestellt
- install: installiert die erstellte Software
- package: erstellt Packages

Bitbake bzw. der gesamte Yocto Build Prozess ist von der Idee her einfach anwendbar, da man sich im Prinzip, sofern verfügbar, einfach ein fertiges Rezept herunterlädt und dieses ausführt. Dies ist jedoch nur dann einfach, solange die Rezepte immer aktuell gehalten werden und keine Fehler enthalten. Gumstix bietet für die Yocto basierte Installation ein eigenes GIT Repository mit einer aus nur wenigen Schritten bestehenden Kurzanleitung an [21], um sich selbst ein Konsolen Image, ein Image mit grafischer Oberfläche oder ein komplett individualisiertes Image, erstellen zu können. Für die ersten beiden Images reicht es aus die zur Verfügung gestellten Rezepte zu verwenden. Für ein individuelles Image ist es jedoch notwendig, sich u.a. erst mit den Rezepten vertraut zu machen, evtl. auch in zig Konfigurationsfiles die der bitbake Prozess mit sich bringt Änderungen vorzunehmen und auch möglicherweise eigene Rezepte zu erstellen.

Auch das Erstellen eines Images nimmt mehr Zeit in Anspruch als das Herunterladen eines vorgefertigten Images. Der Download allein für den Sourcecode sind mehrere Gigabyte. Dies ist jedoch noch relativ schnell erledigt. Der eigentliche Prozess die Dateien zu entpacken, den Sourcecode zu erstellen und alles in die richtigen Verzeichnisse zu kopieren, dauert sehr lange, wenn man nicht gerade einen sehr schnellen Rechner mit zumindest vier bis acht Rechenkernen und genug Arbeitsspeicher sowie Festplattenspeicher zur Verfügung hat. Das Erstellen des Images wurde in einer virtuellen Umgebung, einer VMware mit einem Ubuntu Betriebssystem, durchgeführt. Der virtuellen Maschine wurden zwei Kerne, zwei Gigabyte RAM und genug Festplattenspeicher zugewiesen. Der Prozess vom Initialisieren des Gumstix Repository bis hin zum fertigen Image ging selten schneller als drei Tage. Dies ist vor allem dann problematisch, wenn man den Prozess über Nacht laufen lässt und auf einen Fehler im Rezept stößt, welcher den Prozess dann abbricht. Während man mit der bitbake Option -k das Erstellen des fehlerhaften Rezeptes sozusagen überspringen kann, funktioniert dies natürlich nur bei Rezepten die nicht von dem einen Prozess abhängig sind. Auch das Erstellen des kompletten Images, also alles entsprechend entpacken, in Verzeichnisse kopieren und als .img Datei bereitzustellen, ist auf einer normalen Festplatte auch relativ zeitraubend. Dies hat meist ein bis zwei Stunden gedauert. Die früheren Erstellungsprozesse wurden auf einer SSD durchgeführt, wo dies nur ca. 10 bis 15 Minuten in Anspruch genommen hat. Da jedoch das Verzeichnis ziemlich groß wurde, in Summe mit mehreren Images knappe 300 Gigabyte, reichte der Platz auf einer SSD nicht mehr und musste auf eine normale, langsame Festplatte ausgelagert werden. Glücklicherweise dauert der gesamte Prozess zum Erstellen nicht immer so lange. Ist der Download einmal durchgeführt und sind die Programme erstellt, so sind diese zwischengespeichert und es müssen bei einem Verändern eines Rezeptes nur noch neue Teile heruntergeladen und erstellt werden.

Während des Erstellens der Images sind immer wieder Fehler in Rezepten aufgetreten. Die Probleme der Rezepte konnten eigentlich immer durch (teilweise auch längere) Google-Suchen behoben werden. Im Prinzip waren viele Fehler der Rezepte bereits schon sehr lange bekannt jedoch wurden sie trotzdem nicht korrigiert. Ein möglicher Fehler wäre z. B. einer im Webkit, das zur Erstellung des Images mit grafischer Oberfläche nötig ist. Der Fehler des Rezeptes äußerte sich dadurch, dass der Arbeitsspeicher komplett ausgelastet wurde und selbst mit 13 Gigabyte SWAP immer noch nicht vollendet werden konnte. Erst durch das Beheben des Fehlers im Rezept konnte das Webkit Paket erstellt werden. Gerade bei diesem Paket ist der Fehler recht unangenehm, da das Erstellen davon ca. eine Stunde in Anspruch nimmt und erst kurz vor Ende mit einem Fehler abbricht. So waren mehrere Durchgänge notwendig, um zu sehen ob die Änderungen im Rezept auch Wirkung zeigten. Ähnliche Fehler traten bei mehreren anderen Rezepten ebenso auf. Auch die Verwendung eines anderen Branches des GIT Repository änderte nichts daran - Fehler waren immer vorhanden. Durch diese Erfahrungen sollte relativ schnell ersichtlich sein, warum sich ein selbst erstelltes Image für einen normalen Benutzer auf keinen Fall auszahlt und definitiv nicht

zu empfehlen ist.

Zu Ende des letzten Absatzes wurde jedoch erwähnt, dass es unbedingt nötig ist ein individualisiertes Image zu erstellen. Dies hat den Grund, dass wenn man den DSP des OMAP3530 verwenden will, man den Kernel des Betriebssystems anpassen muss. Dies geschieht durch das Hinzufügen neuer Rezepte, die dann den Kernel entsprechend anpassen und weitere Softwarekomponenten zusätzlich installieren. Bei der Installation wurde sich soweit als an die Anleitungen von Scott Ellis oder [2], [3] gehalten bzw. wurden diese soweit als nötig für die aktuelle Version adaptiert.

Im nachfolgenden Codeabschnitt 3.1 sieht man die Anpassungen, die an das Standard Gumstix Overo Image gemacht wurden. Dabei sind im Prinzip zwei Punkte interessant. Einmal die DSP Kategorie, die alle Pakete für den DSP beinhaltet und einmal die Zusatzsoftware die für die Entwicklung benötigt wird. Als Zusatzsoftware wurde z. B. OpenCV für die Bildverarbeitung und ein C Compiler mit installiert.

```
# TI and OpenCV
TI_OpenCV_INSTALL = " \
 gstreamer-ti \
 gst-plugins-good-meta \
 gst-plugins-base-meta \
 gst-plugins-bad-meta \
 ti-dmai \
 opencv \
 opencv-dev \
 opencv-samples \
 binutils \
 binutils-symlinks \
 cpp \
 cpp-symlinks \
 diffutils \
 file \
 gcc \
 gcc-symlinks \
 g++ \
 g++-symlinks \
 gettext \
 ldd \
 libstdc++ \
 libstdc++-dev \
 libtool \
 make \
 pkgconfig \
"
```

Code 3.1: Ausschnitt aus dem Bitbake Rezept

Das gesamte Rezept zum Erstellen eines Images wird in Anhang A.1 aufgelistet. Es ist zu beachten, dass es durchaus Sinn macht vor dem Erstellen eines angepassten Rezeptes zuerst ein fix fertiges Rezept auszuführen, da dies am stabilsten sein sollte und so ausprobiert werden kann ob der Erstellvorgang erfolgreich ist. Da die Pakete wie zuvor angesprochen gespeichert werden dauert ein Hinzufügen von z. B. DSP Rezepten und ein erneutes Erstellen des Images nicht mehr so lange.

Ist der Erstellvorgang abgeschlossen, erhält man mehrere Files, darunter das erstellte Image, einen Bootloader und ein Kernel Image. Diese sind dann in entsprechender Reihenfolge auf die zuvor korrekt formatierte SD Karte zu kopieren, um anschließend das System booten zu können. Details wie die SD Karte formatiert werden muss und in welcher Reihenfolge die Kopiervorgänge stattzufinden haben sind unter [24] zu finden.

3.2.3 OMAP3530 DSP

Zuvor wurde bereits angesprochen, dass der OMAP3530 Chip, welcher am Gumstix Overo Fire verbaut ist, einen DSP verbaut hat, jedoch wurde noch nicht im Detail darauf eingegangen welche Vorteile dies für diese Arbeit haben sollte. Man kann einfach ausgedrückt den DSP als eigenständigen Prozessor sehen. Das bedeutet nun, dass man sozusagen zwei CPUs hat, einmal den DSP und einmal den ARM Cortex-A8, welcher mit 720 Megahertz getaktet ist während der DSP mit 520 Megahertz betrieben wird. Weiters verfügt der Overo Fire über 512 Megabyte RAM.

Durch den zusätzlich verbauten DSP Chip könnte man nun einen Teil der Berechnungen direkt auf den DSP auslagern oder aber alle Berechnungen auf dem DSP ausführen und nur noch Standardbetriebssystemanwendungen, die nicht für den DSP geeignet sind, auf dem ARM Chip ausführen. Man erhält dadurch natürlich eine bessere Performance und eine höhere Ausführungszeit. Dieser Gewinn geht jedoch leider mit gestiegener Komplexität einher. Dies bedeutet, dass wenn man den DSP wie eine 2. CPU behandelt, sich dies ebenfalls in der Software widerspiegeln muss. Das heißt, man muss auf einige Dinge wie z. B. exklusiven Zugriff auf Daten, Speicherverwaltung und Synchronisation des Ablaufs Rücksicht nehmen. Auch der Overhead muss berücksichtigt werden, da die Daten vom ARM auf den DSP übermittelt werden müssen. Die Daten müssen anschließend am DSP verarbeitet und die Ergebnisse für etwaige weitere Verarbeitung wieder zurück zum ARM transferiert werden. Da der Overhead zum Teil eine nicht so geringe Größe aufweist, sollte nach Möglichkeit immer ein größerer Teil an Berechnungen nacheinander durchgeführt werden, damit der Overhead nicht die gewonnene Rechenleistung zu sehr reduziert.

DSP Prozessoren sind meist für spezielle Anwendungen entwickelt worden. Dabei werden dann spezielle Rechenschritte im Vergleich zu z. B. einem normalen ARM Chip sehr viel schneller ausgeführt. Der Nachteil dabei ist jedoch, dass es nötig ist einen speziellen DSP Code zu schreiben. Man kann also nicht ein beliebiges Programm nehmen und dies ohne weiteres auf dem DSP Kern ausführen lassen. Die Chiphersteller wie Texas Instruments (TI) bieten deswegen oft vorgefertigte Bibliotheken, APIs oder extra erstellte DSP Kernel an, um Zugriff auf den DSP zu erleichtern und so schneller Produkte zu entwickeln. Im Falle von Texas Instruments, welcher Hersteller des OMAP3530 Chips ist, stehen einem eine Vielzahl solcher Bibliotheken zur Verfügung.

In der Tabelle 3.2 sieht man einen Performancevergleich zwischen dem DSP und dem Cortex A8 Chip mit O3 und neon. O3 ist ein Compilerflag und bedeutet, dass beim Compilieren des Quellcodes sehr viele Optimierungen eingeschaltet werden, um so die Codeausführung zu beschleunigen. Andere Stufen wo etwas weniger optimiert wird sind noch O1 und O2 [18]. NEON ist eine Erweiterung des ARM Prozessors und wird ebenfalls eingesetzt um die Ausführungszeiten von Programmen zu senken. Daher tritt der DSP im Benchmark gegen zwei Optimierungsstufen, dem O3 Compilerflag sowie der NEON Erweiterung an.

Wie man anhand der Tabelle 3.2 sehen kann, ist in allen Fällen der DSP trotz der Optimierungen der Ausführung auf dem ARM Chip immer schneller, zum Teil sogar bis zu knapp über 85 Prozent. Dies spricht eindeutig für den DSP und durch den größeren Leistungszuwachs sollte sich

Image Processing Functions	Size of the Image	C6Accel Timing (us)	Cortex A8 with O3+neon	C6Accel improvement over A8
8 bit Histogram	640x480	3266	3387	3.57%
8 bit 3x3 Median filter	640x480	6501	24231	73.17%
8 bit 3x3 signed Image Convolution	640x480	6317	42817	85.25%
8 bit 3x3 Image Correlation	640x480	15351	28992	47.05%
8 bit 3x3 Sobel Edge Detection	640x480	6348	11902	46.66%
8 bit YUV422ILE to YUV422pl Conversion	640x480	16938	30212	43.94%
8 bit YUV422PL to RGB565 Conversion	640x480	16784	20935	19.83%
8 bit Image Addition	640x480	4516	4974	9.21%

Tabelle 3.2: A8 vs. C6Accel (DSP) (nach [10])

der Gumstix Overo Fire mit DSP an sich sehr gut für die Bildverarbeitung auf der ICT Drohne in Kombination mit dem CASPA Camera Modul eignen.

Ohne zu sehr ins Detail zu gehen, wird kurz die von Texas Instruments zur Verfügung gestellte Software zur DSP Entwicklung diskutiert. Eine genauere Darstellung würde den Umfang dieser Arbeit sprengen, da TI zig verschiedene Softwaretools, Wrapper, APIs, DSP Kernels etc. zur Verfügung stellt die teilweise voneinander abhängig sind, teilweise nicht kompatibel zu einander sind oder aber auch nicht mit der HW zusammenarbeiten können. Dem interessierten Leser seien die Whitepaper zu dem jeweiligen Produkt sowie das TI Wiki nahegelegt. Weiters ist zu empfehlen, genau zu suchen (evtl. auch über die Google-Suche), denn selbst wenn man denkt man hat alle Informationen zu dem jeweiligen Produkt gefunden, findet man immer wieder etwas Neues. Die TI Wiki Seite ist extrem umfangreich und verwirrend zugleich.

Als der OMAP3530 Prozessor erschienen ist, war C6EZAccel [11] das aktuelle System zur Entwicklung von ARM + DSP Programmen. C6EZAccel stellt APIs zur Verfügung die nach dem TI spezifischen Algorithmus Interface XDAIS (eXpress DSP Algorithm Interoperability Standard) [54] operieren. Nachfolgend greift C6Accel auf die Codec Engine zu, welche quasi die Schnittstelle zwischen ARM und DSP darstellt. Die Codec Engine greift dann wiederum auf die C6Accel Algorithmen zu.

Eine kurze Übersicht über einige wichtige Libraries ist nachfolgend angeführt. Abbildung 3.10 gibt einen schematischen Überblick, wobei hier auch wiederum darauf zu achten ist, dass natürlich nicht alle Libraries und Codecs eingezeichnet sind. Für eine genauere Übersicht sei auf [17] verwiesen.

- DSP library: z. B. FFT
- Image library: z. B. Farbraumkonvertierungen
- Math library: z. B. Divisionen
- Video Analytics und Vision Library (VLIB): z. B. Lukas Kanade Optischer Fluss
- VoLIB: z. B. Echo Unterdrückung

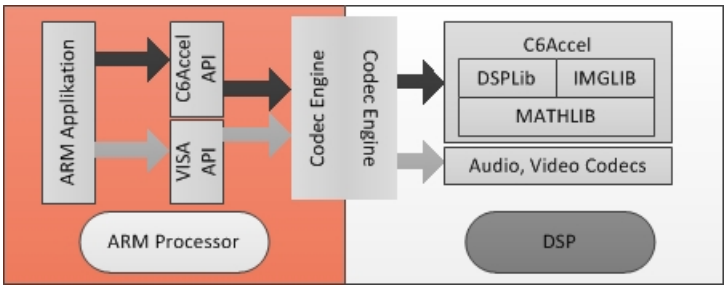


Abbildung 3.10: C6Accel ARM + DSP nach [Pra10]

Während z. B. die Math Library oder VoLIB eher weniger hilfreich für die Umsetzung programmiertechnischer Aufgaben in dieser Diplomarbeit sind, enthält die Video Analytics und Vision Library genau die richtigen Algorithmen. Nämlich die zwei Hauptkomponenten, die für die Hinderniserkennung notwendig sind. Das wäre zum einen ein Feature Tracker, der markante Punkte, die sich ausreichend von anderen Punkten unterscheiden und meist Ecken darstellen, im Feld erkennt. Dazu kommt dann noch der Optische Fluss Algorithmus von Lucas und Kanade, der die bereits zuvor detektierten markanten Punkte über die nächsten Frames verfolgt. Das sind aber nur zwei von vielen Funktionen die in der VLIB vorhanden sind. Andere nützliche Algorithmen wären z. B. Bildpyramiden, welche zum Verkleinern der Bilder, wie bereits in Abschnitt 3.1.3 erläutert, verwendet werden können. Bevor die VLIB entstanden ist, versuchte TI bereits Mitte 2011 direkt OpenCV einzubinden [JC11]. In dem veröffentlichten Dokument waren auch Benchmarks zu finden. Beim Optischen Fluss Algorithmus haben sie eine etwas über 15.2 mal höhere Leistungsfähigkeit des Algorithmus erreichen können wenn dieser am DSP anstatt am ARM ausgeführt wird. Dies wäre natürlich eine sehr schöne Lösung gewesen, da gerade OpenCV den sehr großen Vorteil hat, dass es quasi Plattform-unabhängig ist. Es funktioniert auf Linux, Windows, MAC, Android und iOS. Dazu auf den verschiedensten Hardwareplattformen wie üblich auf X86 PCs aber auch auf ARM Chips, die entweder wie hier auf dem Gumstix Overo Fire oder als neuere und leistungsstärkere Version in Mobiltelefonen eingesetzt werden. Diese Freiheiten wären aus Entwicklersicht natürlich ideal. Man könnte dann die Applikation am PC entwickeln mit allen mächtigen Debugging Tools die zur Verfügung stehen und kann das fertige Programm dann ohne weiteres auf einem sehr kleinen, stromsparenden Gerät wie dem Gumstix Overo Fire laufen lassen. Die genauen Gründe warum TI die OpenCV Portierung eingestellt hat, sind leider nicht bekannt. In einem Post im offiziellen Texas Instruments Forum hat ein Mitarbeiter geschrieben, dass nicht mehr an einer Portierung von OpenCV für den DSP gearbeitet wird [39]. Die Vermutung liegt nahe, dass sie Kunden auf ihrer Plattform halten wollen. Denn wenn man den Code extra für den ARM+DSP schreibt, ist ein Portieren doch etwas aufwendiger als wenn man Plattformübergreifend OpenCV verwenden würde. Im Rahmen des "opencv-dsp-acceleration"Projektes [40] versuchte man eigenständig ohne Hilfe von TI OpenCV Funktionen für den DSP zu portieren. Dies war bei den ersten Benchmarks nicht erfolgreich. Bei einer späteren Version des Programmes wurden erneut Benchmarks veröffentlicht, jedoch können die zur Verfügung gestellten Werte nicht vollständig nachvollzogen werden. Das Projekt wurde bereits wieder eingestellt und hat das letzte Update im August 2010 erfahren.

Da Texas Instruments die Weiterentwicklung von OpenCV eingestellt hat, muss man also auf die hauseigenen Libraries setzen um den DSP verwenden zu können. Zu Beginn dieser Arbeit war für die VLIB (Version 2.x) noch eine extra Anforderung nötig mit der man für den Download freigeschaltet worden ist, während für die anderen Libraries direkter Zugriff gewährt wurde. Selbst

nach zwei Monaten Wartezeit und mehreren Anforderungen wurde kein Zugriff gewährt. Mit Stand März 2014 ist es jedoch möglich die VLIB bereits in Version 3.x direkt zu beziehen, ohne extra eine Anforderung stellen zu müssen.

3.2.4 GStreamer - Video Übertragung

GStreamer ist ein open source Multimedia Framework [19]. In dieser Arbeit wurde zu Beginn GStreamer verwendet um Bilder bzw. einen Videostream des am Gumstix Board angesteckten CASPA Camera Moduls über WLAN auf einen PC zu übertragen. Der Gedanke dahinter und die Gründe warum man ein Video vom Gumstix Overo Fire zum PC übertragen sollte werden kurz angeführt:

- Videoübertragung ermöglicht Fernsteuerung auch ohne direkten Sichtkontakt
- Videoübertragung würde die Ausführung und Berechnung der Hinderniserkennung am PC erlauben, um anschließend Steuerbefehle zurück zur Drohne zu senden
- Man kann direkt mit der realen Kamera arbeiten, auch wenn man noch keinen Zugang zu der VLIB hat

Für GStreamer stellt TI ein Plugin zur Verfügung damit auf den DSP zugegriffen werden kann. Dies ist essentiell um eine gute Performance zu erreichen. Während sich bei Versuchen mit anderen Programmen wie z. B. mjpeg-streamer herausgestellt hat, verursachen diese 100% CPU Auslastung ohne eine annehmbare Framerate zu erzielen. GStreamer hingegen konnte in dieser Hinsicht durch den Einsatz des DSP punkten. Um GStreamer und die CASPA Camera jedoch zuerst funktionsfähig zu bekommen, müssen nicht nur die Treiber installiert werden sondern es muss auch das media-ctl Framework richtig konfiguriert werden. Ohne die Konfiguration mit dem media-ctl Framework würde die Kamera nicht wie gewünscht funktionieren. Das media-ctl Framework wurde mit v4l2 eingeführt. Nachfolgend werden die zwei Zeilen, die für die Konfiguration benötigt werden, angezeigt. Auch wenn es auf den ersten Blick recht einfach aussehen mag, bestätigt sich diese Annahme auf den zweiten Blick nicht. Durch die vielfältigen Einstellungsmöglichkeiten die das media-ctl Framework bietet, können ganz leicht Fehler in der Konfiguration auftreten. Der Syntax, wie man in 3.2 sieht, ist auch nicht unbedingt der Übersichtlichste. Man verbindet im Prinzip die einzelnen Ebenen, wie z. B. ISP CCDC, ISP preview oder auch ISP Resizer und kann auch verschiedene Ausgabeformate der einzelnen Ebenen bestimmen, z. B. SRGB10 mit einer Auflösung von 752x480, das dann in UYVY konvertiert wird und in 640x480 ausgegeben wird. Die Auflösung von 640x480 ist nötig, da der TI TIVidenc1 nicht mit einer Auflösung von 752x480 zurecht kam und nur diverse Fehlermeldungen ausgeben hat. Diese Fehler konnten nicht gelöst werden, darum wurde die Auflösung auf 640x480 reduziert.

```
media-ctl -r -l '"mt9v032 3-005c":0->"OMAP3 ISP CCDC":0[1],
                "OMAP3 ISP CCDC":2->"OMAP3 ISP preview":0[1],
                "OMAP3 ISP preview":1->"OMAP3 ISP resizer":0[1],
                "OMAP3 ISP resizer":1->"OMAP3 ISP resizer output":0[1]'

media-ctl -V '"mt9v032 3-005c":0[SGRBG10 752x480],
             "OMAP3 ISP CCDC":2[SGRBG10 752x480],
             "OMAP3 ISP preview":1[UYVY 752x480],
             "OMAP3 ISP resizer":1[UYVY 640x480]'
```

Code 3.2: Camera Setup via media-ctl

Ist das media-ctl Framework nun soweit konfiguriert, dass die CASPA Kamera die Bilder im richtigen Format und in der richtigen Auflösung liefert, kann mit dem GStreamer die eigentliche Komprimierung sowie Übertragung des Videostreams erfolgen. Eine Beispielbefehlszeile 3.3 ist nachfolgend angeführt.

```
LD_PRELOAD=/usr/lib/libv4l/v4l2convert.so \
gst-launch -e -v v4l2src device=/dev/video6 always_copy=FALSE ! \
'video/x-raw-yuv,width=640,height=480,format=(fourcc)UYVY' ! \
TIPrepEncBuf contiguousInputFrame=FALSE ! queue ! \
TIVidenc1 codecName=h264enc engineName=codecServer ! \
dmaiperf print-arm-load=true engine-name=codecServer ! \
rtph264pay pt=96 ! udpsink host=10.0.0.8 port=5000 --gst-debug-level=2
```

Code 3.3: GStreamer Befehl zum Komprimieren via H264 sowie Streamen des Videobildes an einen Empfänger über das UDP Protokoll

Zuerst muss das v4l2convert Modul geladen werden, anschließend erfolgt der GStreamer Aufruf. Dabei wird u.a. die Videoquelle, in diesem Fall /dev/video6, die Auflösung und das Videoformat festgelegt. Ebenso wird der TIVidenc1 verwendet, dieser ist für die h.264 Komprimierung des Videos via DSP zuständig. Gerade die Komprimierung ist wichtig, da WLAN vor allem bei größeren Distanzen eine eher geringe Netto-Bandbreite aufweist. Ein h.264 kodiertes Video mit einer Aufnahme weche mit der CASPA Kamera durchgeführt wurde hat eine Bitrate von etwa 1700Kbit/s. Ein unkomprimiertes Video benötigt ein vielfaches dessen, in etwa 10 Mbit/s. Wenn man davon ausgeht, dass die maximale WLAN Nettobandbreite maximal in etwa 13Mbit beträgt kann es sein, dass es zu Bildfehlern aufgrund von zu wenig Bandbreite kommt. Anschließend wird der Stream über UDP übertragen, in diesem Fall an die IP 10.0.0.8. Der Teil *dmaiperf print-arm-load=true engine-name=codecServer* des Aufrufes ist besonders interessant da es mit diesem Aufruf möglich ist, die Performance zu evaluieren und die Last sowohl von der CPU als auch vom DSP auszulesen. Eine beispielhafte Ausgabe des dmaiperf Befehles ist in 3.4 angeführt.

```
Timestamp: 0:05:25.070911190; bps: 12340533; fps: 25; CPU: 22; DSP: 83;
mem_seg: DDR2; base: 0x8fe8b1e6; size: 0x20000; maxblocklen: 0x15030; used: 0x9ec0;
mem_seg: DDRALGHEAP; base: 0x88000000; size: 0x7a00000; maxblocklen: 0x7860d00;
        used: 0x24e400;
mem_seg: L1DSRAM; base: 0x12e43000; size: 0x10000; maxblocklen: 0x800; used: 0xf800;
```

Code 3.4: Beispielhafte Ausgabe von dmaiperf

Während der Tests war die beste zu erzielende Framerate knapp 25 Frames per Second (FPS) bei einer CPU Last von nur 15-20 Prozent und einer DSP Last von 80 Prozent. Der Aufruf der GStreamer Kommandos ist wichtig und erfordert etwas Zeitaufwand um dies zu optimieren. Anfangs waren nur knapp 15 FPS möglich und mit ein paar zusätzlichen Argumenten wie z. B. *always_copy=FALSE* konnte dies auf die angesprochenen 25 FPS erhöht werden. GStreamer bietet extrem umfangreiche Möglichkeiten und bietet sich auch für viele andere Arbeiten die mit Videostreaming zu tun haben an.

Damit alles wie gewünscht funktioniert, waren jedoch nicht nur Treiberanpassungen nötig, sondern es mussten auch diverse Patches händisch eingespielt und dann die einzelnen Module neu compiliert sowie das Image neu erstellt werden. Mit GStreamer hat das Streamen ausreichend gut funktioniert, jedoch die Möglichkeiten andere Software, u.a. die Bildverarbeitungslibrary OpenCV zu verwenden, war nicht möglich. Dies war laut den Fehlern, die OpenCV ausgegeben hat, und einer zusätzlichen Recherche dem etwas spezielleren Ausgabeformat sowie dem V4L2 Treiber geschuldet. Es war nicht möglich, die CAPSA Camera in Verbindung mit OpenCV lauffähig zu bekommen. Es wäre soweit man das beurteilen kann nötig gewesen, noch tiefer gehende Eingriffe und spezielle Anpassungen für die CASPA Camera im Treiber zu machen. Auch wird von TI so gut wie kein Support mehr für die Plattform bereitgestellt, wodurch dieses System für zukünftige Erweiterungen Einschränkungen mit sich bringt.

3.3 Reevaluierung einer neuen Hardware Plattform

In diesem Kapitel wird die Reevaluierung einer neuen Hardware Plattform für die Drohne behandelt, da dies aus mehreren bereits angeführten Gründen nötig ist. Dies wären z. B. kein Zugriff zur VLIB, die alleine nicht ausreichende Leistung des ARM Chips, sowie Probleme mit dem CASPA Camera Modul. Diese Probleme haben es nötig gemacht, dass man sich von der vorhandenen Plattform zumindest zum Teil trennt bzw. diese für andere Aufgaben einsetzt. Während die Steuerung noch beim ursprünglichen Zustand, also beim ATMEGA Prozessor sowie allen Sensoren, wie etwa Lagesensoren die an diesem Prozessor hängen sollen, erhalten bleibt.

3.3.1 State of the Art ARM Plattformen

Es soll also eine neue Hardware Plattform gefunden werden, die alle Probleme der alten Plattform so weit wie möglich behebt und gleichzeitig mehr Leistung bringt - also so gesehen eine State of the Art ARM Plattform. Als Anforderung an die neue Plattform können folgende Punkte definiert werden:

- Gewicht sollte gering sein, maximal 200-300 Gramm
- Ausmaße so, dass die neue Plattform gut auf der Drohne verbaut werden kann
- deutliche Leistungssteigerung gegenüber der alten Plattform
- für zukünftige Erweiterungen gerüstet sein, d.h. genug Arbeitsspeicher und zumindest ein Dual oder Quadcore Prozessor wären wünschenswert um z. B. mehrere Algorithmen zugleich auszuführen und/oder die Leistung durch paralleles Ausführen dieser zu steigern
- ein DSP, der wie beim Gumstix Overo Fire die Bildverarbeitung effizienter als ein ARM Prozessor durchführt, sollte vorhanden sein
- die Kosten sollten im Rahmen bleiben
- gute Entwicklungsumgebung inkl. Debugger
- wenig Stromverbrauch um die Flugzeit nicht noch mehr zu verkürzen
- Möglichkeiten weitere Peripherie zu verwenden

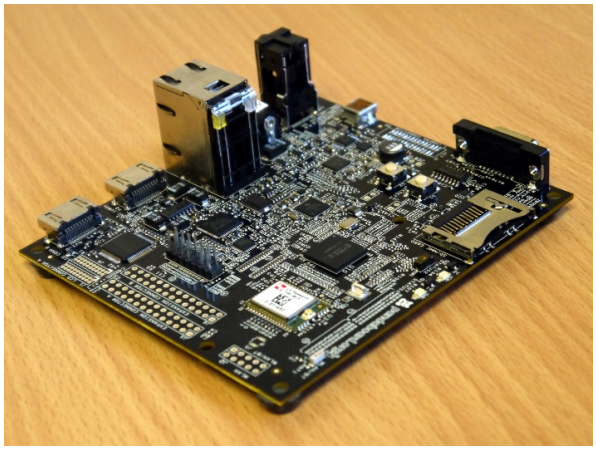
- Möglichst Plattform-unabhängig

Anhand der Liste der Anforderungen sollte eine möglichst passende neue Hardware Plattform gefunden werden. Wenn es nicht möglich ist eine Plattform zu finden, die exakt diese Anforderungen erfüllen kann, sollen diese zumindest soweit wie möglich angenähert werden. Da Plattformen, basierend auf Standard PC Komponenten, zu viel Strom und Platzbedarf benötigen und auch vom Gewicht her zu schwer sind, wurde der Fokus auf Embedded Systeme, die am Markt erhältlich sind, gelegt.

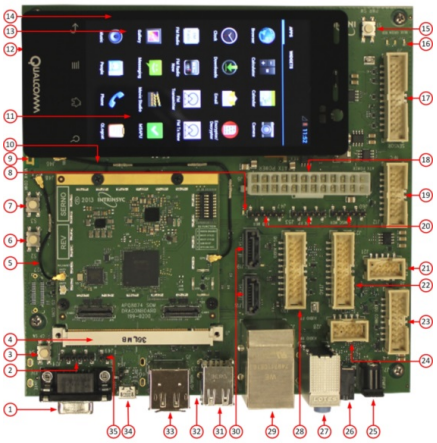
Der Gedanke liegt natürlich nahe einen Nachfolger des Gumstix Boards zu verwenden, da der Gumstix Overo Fire an sich eine sehr gute Basis für den Hauptcomputer der Drohne darstellt, der nur bereits etwas in die Jahre gekommen ist und auch die Probleme mit dem CASPA Camera Board hat. Von Gumstix ist der Overo FireSTORM [20], welcher auf einem OMAP3570 Chip basiert das leistungsfähigste Modell. Dieser bietet eine Leistung von 1 Gigahertz ARM Cortex A8 (wobei nur 800 empfohlen werden), 1 Gigabyte RAM und verfügt genau wie der Overo Fire über einen Fixed Point DSP und kostet laut Gumstix Webseite umgerechnet knapp 145 Euro. Der Gumstix Overo FireSTORM eignet sich also nicht als Nachfolger oder als Zusatz für den Overo Fire. Als direkte Konkurrenz, teilweise sogar auf dem selben Chip basierende, Hardware Plattformen stellen z. B. das Beagleboard bzw. das neuere BeagleBoard Black dar. Das Beagleboard Black, welches das neueste Produkt der BeagleBoard Serie ist, entspricht von den Leistungsdaten her in etwa dem Gumstix Overo FireSTORM, was bedeutet, dass dieses Board auch zu wenig Leistung aufweist. Ein weiteres interessantes Embedded System stellt das Pandaboard [43] dar, siehe Abbildung 3.11 (a). Im Vergleich zu den bereits diskutierten Vertretern von Embedded Systemen basiert dieses bereits auf einem ARM Cortex A9 statt einem Cortex A8. Der ARM Cortex A9 Prozessor unterscheidet sich vom A8 betreffend der Leistung, aber auch vom Umstand her, dass der auf dem Pandaboard verbaute Chip bereits ein Dual Core Chip ist. Außerdem besitzt das Pandaboard einen DSP. Bei der Recherche hat sich jedoch ergeben, dass der am Pandaboard verbaute DSP einerseits langsamer ist als der auf dem Overo Fire und andererseits seitens Texas Instruments (welche den Chip fertigen) noch weniger Zugriff darauf möglich ist. In Summe bleibt beim Pandaboard also nur die etwas schnellere CPU und etwas mehr Arbeitsspeicher als Pluspunkt gegenüber der derzeitigen Plattform.

Als leistungsfähigere Vertreter von Embedded Systemen wäre hier ein Quad Core System, basierend auf einem Samsung Exynos Chip der mit 1.4 Gigahertz getaktet ist, das so genannte Origen Board [41] anzuführen. Dieses Board bietet nicht nur eine Quadcore CPU, sondern es sind auch Expansionsmodule in Form von einem Kamera Modul, einem GPS Modul oder auch einem Expansionsboard welches einen Beschleunigungsmesser und Gyroscope verbaut hat, verfügbar. Ein im Vergleich zum Origen Board noch leistungsfähigeres Gerät ist das Dragonboard [15]. Das Dragonboard verfügt über einen überaus leistungsfähigen Snapdragon 800 Prozessor von Qualcomm zusätzlich zu einem, für ein Embedded System üppigen 2 Gigabyte Arbeitsspeicher, und bietet daher fast alles was man nur brauchen kann, wie etwa USB 2.0 und 3.0 Anschlüsse, SATA, GBIT Ethernet oder JTAG Header zum Debuggen. Auch ein sogenannter Hexagon DSP, welcher am Snapdragon 800 Chip verbaut ist, ist verfügbar. Das Dragonboard hat jedoch zwei größere Kritikpunkte: Einerseits der Formfaktor, da das Dragonboard in Mini ITX Bauform ausgeführt ist und andererseits der Preis von derzeit ca. 380 Euro. Glücklicherweise ist der Snapdragon 800 ein sehr weitverbreiteter Chipsatz. Dieser wird nämlich in so gut wie allen derzeit aktuellen High End Android Smartphones, wie dem Nexus 5 oder auch dem LG G2, verbaut.

Der Vorteil ein modernes Smartphone als Hauptcomputer oder zumindest für die Hinderniserkennung der Drohne einzusetzen liegt auf der Hand. Handys sind je nach Ausführung billig,



(a) Pandaboard. Abbildung von Embecosm Limited nach CC BY-SA 2.0 [42]



(b) Dragonboard. Abbildung von [16] [1]

Abbildung 3.11: Ausschnitt aus einer Vielzahl an verfügbaren Embedded Boards

gerade Android Smartphones leiden unter einem sehr hohem Preisverfall bzw. wenn man auf ein neues Smartphone umsteigt bleibt ein altes oft über. Daraus folgt, dass man auf breiter Basis relativ günstig Android Smartphones, z. B. auch gebrauchte Geräte, erwerben und diese dann für die Hinderniserkennung einsetzen kann. Es müssen jedoch nicht unbedingt die neuesten Highend Smartphones mit Snapdragon 800 Chip sein, günstigere der 400er und 600er Serie haben ebenfalls den leistungsstarken Hexagon DSP verbaut. Weiters bringt der Einsatz von Android als Betriebssystem zusätzliche Vorteile mit sich, die berücksichtigt werden sollten. Dies wären z. B. eine gute Entwicklungsumgebung sowie gute Möglichkeiten zum Debuggen der Anwendung. Auch kann OpenCV durch die Plattform-übergreifende Kompatibilität verwendet werden. Ein Smartphone aus der höheren Preisklasse verfügt außerdem über viele Sensoren wie ein Gyroskop, einen Beschleunigungssensor und einen digitalen Kompass. Natürlich verfügt das Smartphone auch über GPS sowie über eine Kamera, die mit einer hohen Auflösung und einer guten Optik, ausgestattet ist. Wie man also sieht, deckt ein Smartphone die derzeitigen Anforderungen an ein Embedded System gut ab und bietet mit z. B. GPS und 3G oder LTE Modem sowie dem USB Host Modus noch die Möglichkeit für zukünftige Erweiterungen. Auch wenn derzeit Smartphones mit einem Chip aus der Qualcomm Snapdragon Serie weit verbreitet sind, gibt es noch einige wenige andere die einen Samsung Exynos Chip wie das Origen Board oder einen Chip von NVIDIA verbaut haben. Der Exynos Chip ist aufgrund des fehlendem DSPs und subjektiv schlechterer Dokumentation zur Entwicklung weniger empfehlenswert bzw. geeignet, während der NVIDIA Chip durchaus interessant wäre. Da jedoch zum Stand dieser Arbeit fast alle leistungsstärkeren Smartphones auf dem Qualcomm Chip basieren, schließt dies leider derzeit die Nvidia Plattform aus. Gerade zum Entwickeln wäre diese jedoch auch sehr gut geeignet. Dies hat den Grund, dass OpenCV und NVIDIA zusammenarbeiten. NVIDIA bietet mit dem Tegra Android Development Pack [53] die Möglichkeit ausgewählte OpenCV Algorithmen ohne irgendwelche Anpassungen am Quellcode und ohne zusätzlichen Aufwand bei der Programmierung die Hardware beschleunigt zu verwenden. NVIDIA hat ein paar Beispielfunktionen sowie deren Performancegewinn angegeben. Die Funktion median blur ist z. B. bis zu 23.7 mal so schnell und der Optische Fluss bis zu 1.5

[1] In der Abbildung sind Nummerierungen für die einzelnen Ports zu sehen. Die Beschreibung zur Nummerierung ist in der Quelle der Abbildung zu finden.

mal so schnell als ohne Hardwarebeschleunigung.

3.3.2 FastCV und OpenCV

FastCV und OpenCV spielen in dieser Diplomarbeit bei der praktischen Implementierung der Hinderniserkennung auf der entsprechenden Hardware eine wichtige Rolle . FastCV ist eine von Qualcomm zur Verfügung gestellte Bildverarbeitungslibrary, die auf sehr vielen Plattformen verfügbar ist. Programme können für Android Smartphones, Windows RT, Windows Phone 8 entwickelt werden und auch zum Erstellen der Programme kann auf mächtige Entwicklungsumgebungen wie etwa Visual Studio zurückgegriffen werden. Auch eine Entwicklung auf Linux und Mac OSX ist ebenso möglich. FastCV deckt also alle derzeit gängigen Plattformen, bis auf Apples eigenes Smartphone Betriebssystem iOS, ab. Zwar bietet FastCV nicht so viele Bildverarbeitungsfunktionen bzw. Algorithmen wie OpenCV, jedoch sind einige wichtige wie der Optische Fluss Algorithmus von Lucas und Kanade sowie der FAST Algorithmus zur Erkennung markanter Punkte vorhanden. Das wirklich Schöne an FastCV und OpenCV ist, dass diese beiden parallel sowohl auf einer Standard PC Plattform, als auch auf einem z. B. Android Smartphone, laufen. Das bedeutet konkret, dass einem alle Funktionen von OpenCV zur Verfügung stehen und auch alle von FastCV, d.h. sollte es einen Algorithmus nicht in der FastCV Library geben, kann man einfach den von OpenCV verwenden. Ein kleiner Nachteil muss jedoch erwähnt werden, es ist zum Teil nötig, dass die Daten in ein anderes Format konvertiert werden, da die FastCV API mit anderen Variablentypen arbeitet als OpenCV. Dieser Nachteil wird aber durch die erhöhte Performance, die FastCV bietet bei weitem ausgeglichen. Auch kann man z. B. sein Programm komplett in OpenCV schreiben, die Performance analysieren und dann einfach Funktionen, die eine lange Ausführungszeit haben, mit Funktionen (sofern diese vorhanden sind) von FastCV austauschen. Welche Performance von FastCV erreicht werden kann, wird im Punkt "Performance Evaluierung"genauer untersucht. Zwei weitere wichtige Punkte, ein positiver sowie ein negativer, sind hier noch zu erwähnen. Positiv ist, dass FastCV die Möglichkeit bietet, den Hexagon DSP, der in den Qualcomm Snapdragon Chips verbaut ist, zu verwenden. Der negative Punkte ist der, dass die Dokumentation, sowie das Qualcomm Hilfe Forum zu FastCV etwas spärlich ausfallen. Es ist daher oft nötig etwas auszuprobieren, um die Funktion auch so anwenden zu können, wie sie anzuwenden ist. Die Kombination aus OpenCV und FastCV ist bereits ein relativ guter, wenn auch noch nicht zu 100% idealer, Ansatz, da er immer noch, wenn auch nicht in ganz so starkem Ausmaß, für eine Plattform optimiert werden muss. In Zukunft sollte dies aber hoffentlich durch die Bemühungen von Khronos mit der OpenVX hardware acceleration API [28] nicht mehr nötig sein. OpenVX hat das Ziel, dass man alle möglichen Hardwarebereiche wie CPU, GPU, DSPs oder sonstige Hardware, die in modernen mobilen Systemen zur Verfügung steht, abstrahiert und OpenVX bzw. dem Hersteller überlässt, welches Hardwaremodul am besten zum Ausführen der bestimmten Aufgabe geeignet ist. In Abbilung 3.12 wird dieser Zusammenhang noch einmal grafisch dargestellt. Derzeit befindet sich OpenVX noch in der Entwicklung, mit einer finalen Spezifikation wird für Mitte 2014 gerechnet. Es sind nahezu alle namhaften Hersteller, wie etwa Qualcomm, Texas Instruments, Intel, ARM und noch viele weitere Branchengrößen, mit an Board. Sollte dies so verwirklicht werden wie geplant, ist dies für die Entwicklung von Programmen sehr zu begrüßen. Man schreibt dann sein Programm einmal und es läuft auf jeder Plattform die OpenVX unterstützt (was nach den Partnern zu urteilen sehr viele sein werden) mit der bestmöglichen Leistung unter Ausnützung der diversen Hardwarefeatures die in dem jeweiligen Gerät verbaut sind.

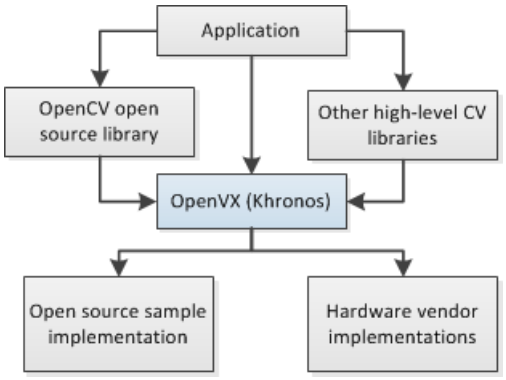


Abbildung 3.12: OpenVX hardware acceleration. Dadurch wird ermöglicht, dass High Level Libraries, wie OpenCV auf die Implementierung gewisser Funktionen von Herstellern zugreifen. Dadurch wird es möglich, die Ausführungen zu beschleunigen, da die Hersteller von Hardware, wie im falle von Qualcomm auch z.B. den DSP für den Algorithmus verwenden können. Durch Khronos bzw. OpenVX soll das herstellerübergreifend funktionieren. Abbildung nach [28]

3.3.3 Performance Evaluierung

Um ungefähr zu sehen und beurteilen zu können, wie die Performance von FastCV und OpenCV auf einem modernen Smartphone ist, sollte bevor eines angeschafft wird auch sichergestellt werden, dass dieses die Anforderungen an die Laufzeiten des Algorithmus erfüllen kann. Auch soll nicht nur ein kleiner Performance-Fortschritt, sondern eine deutlich größere Leistung zur Verfügung stehen als mit dem Gumstix um für zukünftige Erweiterungen so gut wie möglich gerüstet zu sein. Auch schreibt Qualcomm leider relativ wenig über die wirklichen Performance gewinne beim Einsatz von FastCV. Auf der Qualcomm Webseite steht lediglich, dass in einer Applikation die bereits fertig entwickelt war, auf FastCV umgestiegen wurde und diese in nur etwa zwei Tagen Entwicklungszeit eine Performancesteigerung von 15 Prozent zu verzeichnen hatte. In Anbetracht von nur zwei Tagen zusätzlicher Entwicklungszeit ist der Gewinn an Leistung beachtlich. In der Präsentation von Yossi Cohen [Coh12] ist die Performance für einige Algorithmen näher aufgeschlüsselt. Dabei wird auch die Performance von FastCV für ARM und FastCV für Snapdragon verglichen und in Abbildung 3.13 dargestellt.

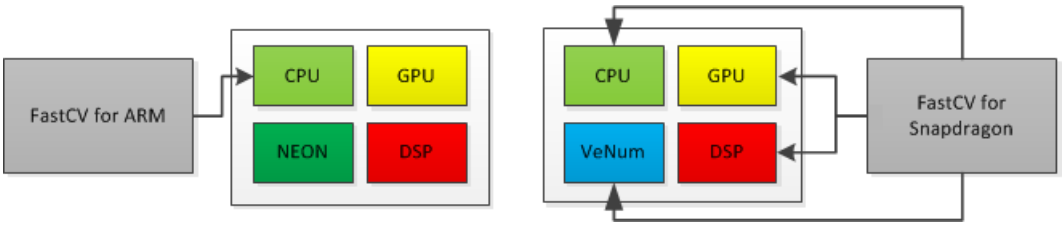


Abbildung 3.13: FastCV Implementierung - wenn die entsprechende Hardware verfügbar ist wird "FastCV für Snapdragon" verwendet welche u.a. auf den DSP Chip zugreifen kann, ansonsten wird nur die normale ARM Implementierung ausgeführt, die aber zumindest über eine NEON Optimierung verfügt welche der Ausführungszeit ebenfalls zu gute kommt. Abbildung nach [13].

Sollte FastCV auf einer nicht Snapdragon Plattform ausgeführt werden, stellt dies kein Problem

dar, da das Programm ebenso ausgeführt werden kann, nur mit dem Unterschied, dass dieses nicht mehr für die Snapdragon Plattform optimiert ist. Das bedeutet einfach ausgedrückt, dass alles über die CPU gerechnet werden muss und nicht mehr auf die spezifische Hardware die in einem Snapdragon Chip verbaut ist zugegriffen werden kann. Trotzdem soll die Performance selbst mit der Verwendung von FastCV für ARM besser sein als nur die OpenCV Implementation des auszuführenden Algorithmus. In Tabelle 3.3 werden die erreichten Werte der Algorithmen aufgeschlüsselt.

Function	OpenCV	FastCV	FastCV Snapdragon
NCC	1.0x	9.0x	23.1x
Dot Product 128x4	1.0x	4.0x	10.0x
Convert YUV420	1.0x	1.4x	1.3x
Sobel	1.0x	1.8x	7.8x
Median3x3	1.0x	3.8x	51.9x
Gaussian3x3	1.0x	2.6x	4.1x
Gaussian5x5	1.0x	1.4x	2.9x
Threshold	1.0x	0.7x	9.7x
Integral Image	1.0x	1.1x	1.3x
Harris Corner	1.0x	2.8x	8.6x
Dilate	1.0x	1.4x	15.0x
Erode	1.0x	1.3x	15.0x
Perspective Fit	1.0x	21.5x	37.8x
LK Optical Flow	1.0x	2.0x	14.3x

Tabelle 3.3: Vergleich OpenCV vs FastCV für ARM und FastCV für Snapdragon

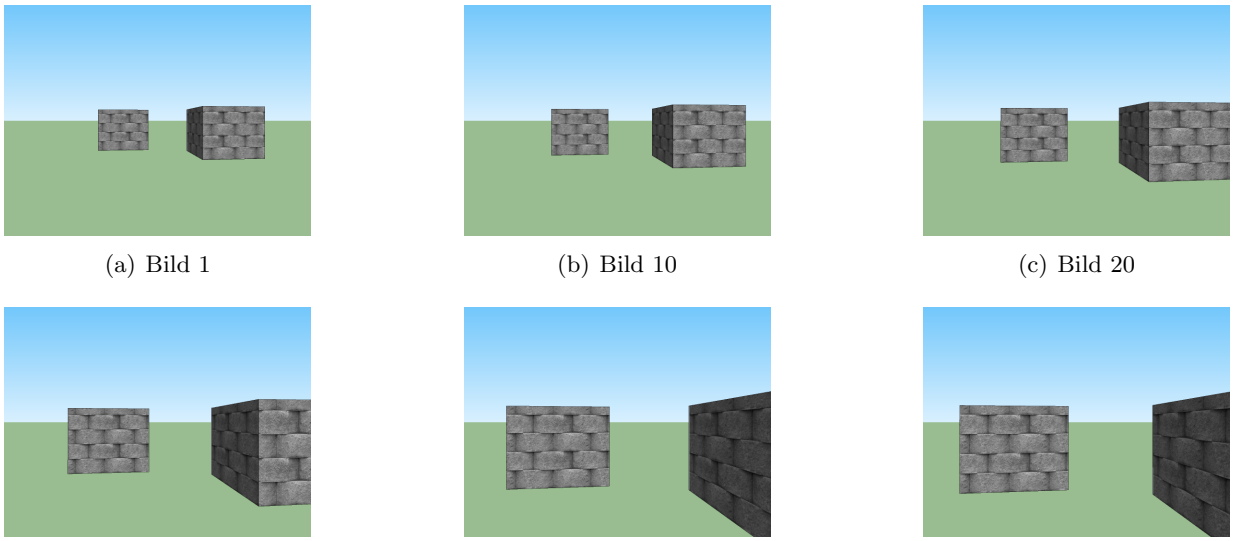
Der Vorteil, Berechnungen zusätzlich z. B. auch am DSP laufen zu lassen, ist aber gerade bei mobilen Anwendungen der, dass die Akkulaufzeit verlängert werden kann. Im Falle der Drohne ist dies nicht unbedingt relevant, wenn man eine Flugzeit von etwa 7-10 Minuten mit der Laufzeit eines modernen Smartphones vergleicht. Selbst wenn das Smartphone alle Akkusparmodi wie z. B. Displayhelligkeitsregulierung, Module wie z. B. WLAN nicht abschaltet wird und auch ein extrem leistungshungriges Programm ausgeführt wird, wird die Akkulaufzeit die Flugzeit übertreffen. Sollte das Smartphone eine Stromversorgung mit der Embedded Plattform teilen, so sollte das Smartphone auch einen geringen Anteil am Gesamtstromverbrauch haben da gerade Mobile Geräte sehr auf einen geringen Stromverbrauch getrimmt sind und große Verbraucher wie z. B. das Display auf einer Drohne natürlich ausgeschaltet werden kann da dies nicht benötigt wird.

Um also die reale Performance zu evaluieren wurde schrittweise folgendermaßen vorgegangen:

1. Erstellen einer synthetischen Videosequenz
2. Erstellen eines Android JNI Programmes, einmal wird der Code komplett mit OpenCV ausgeführt, einmal komplett mit FastCV Funktionen
3. Verteilen des entwickelten Demo Programmes an verschiedene Personen mit verschiedenen Android Versionen und Smartphones bzw. Tablets
4. Auswerten der via Mail erhaltenen Ergebnisse

Erstellen einer synthetischen Videosquenz

Zum Erstellen einer synthetischen Videosequenz wurde Trimble Sketchup (bekannt durch Google, war früher auch im Besitz von Google) verwendet. Sketchup bietet die Möglichkeit schnell und einfach Videosequenzen zu erstellen. Es kann auch auf das 3D Warehouse zurückgegriffen werden, in dem sich tausende fertige 3D Modelle in den verschiedensten Kategorien befinden. Die erstellte Testsequenz sollte jedoch möglichst einfach gehalten werden. Aus diesem Grund wurden nur zwei Würfel erstellt, die in einem Abstand versetzt zueinander angeordnet wurden. Zusätzlich erhielten die Würfel eine Steintextur, da ohne Textur eine Erkennung von markanten Punkten fast nicht mögliche ist. Die Schwierigkeiten von Kamera Systemen bei wenig oder gleicher Textur wurde bereits in Kapitel 2 ausgiebig diskutiert. In Abbildung 3.14 sieht man einige Frames der Videosequenz, die aus insgesamt 121 Bildern besteht. Die Kamerabewegung in der Videosequenz ist eine reine Vorwärtsbewegung, also Translation ohne einen Rotationsanteil.



(a) Bild 1 (b) Bild 10 (c) Bild 20

(d) Bild 30 (e) Bild 40 (f) Bild 80

Abbildung 3.14: Videosequenz der Szene im Evaluierungsprogramm

Erstellen eines Android JNI Programmes

Da die Performance im Simulator bzw. Emulator im Vergleich zu realen Geräten sehr schlecht ist und dadurch wenig Aussagekraft hat, soll die Performance natürlich soweit wie möglich unter realen Bedingungen auf modernen Smartphones getestet werden. Dazu war es notwendig, ein Android Programm zu erstellen, das sowohl die Implementierung der Evaluierung, also des Optischen Fluss Algorithmus, als auch die Erkennung markanter Punkte durch den FAST Algorithmus umsetzt. Android Programme werden normalerweise in Java geschrieben. Durch das Java Native Interface (JNI) wird es möglich, die Programme auch in C zu schreiben. Dabei wird die GUI etc. weiter als Java Programm behandelt, aber es erfolgt ein Aufruf zum C Programm. OpenCV ist auf der Android Plattform sowohl für Java als auch für C verfügbar. Es wird jedoch empfohlen, die C Umgebung anstatt der Java Umgebung zu verwenden - eben durch die Möglichkeit der JNI Calls.

Einfach ausgedrückt kann man sich die JNI Calls wie Funktionsaufrufe aus Java heraus vorstellen. Durch die Möglichkeit Android Programme auch in C zu schreiben, kann man bis auf z. B.

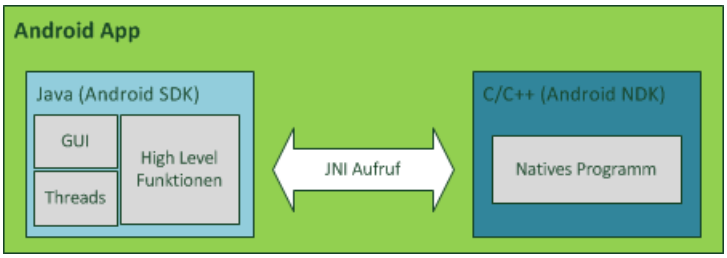


Abbildung 3.15: JNI Call

Anpassungen bei der Ausgabe von Logging Informationen (Android verwendet ALOG während bei C z. B. diese mit cout auf die Standardausgabe geschrieben werden) den Code der zuvor z. B. in Visual Studio in C geschrieben wurde in der Android App verwenden.

Beim Programm wird die komplette Ausführungszeit gemessen. Diese beinhaltet alle Variablen, Deklarationen, Funktionsaufrufe etc. In Summe werden vier Testläufe durchgeführt, zwei mit FastCV und zwei mit OpenCV wobei jeweils einmal die zuvor erstellte Videosequenz mit 121 Frames, sowie einmal nur zwei Bilder der gleichen Sequenz durchlaufen werden. Die Testsequenz bei der nur zwei Bilder verglichen werden ist deshalb notwendig, da bei der Videotestsequenz, durch z. B. Trackingfehler oder wenn die Features nicht mehr im Bild sichtbar sind, die Features über die Zeit immer weniger werden. Beim richtigen Programm ist es daher unbedingt erforderlich eine Grenze einzuführen ab der wieder neue Features mit dem FAST Algorithmus gesucht werden die dann für das Tracking mit dem Optischen Fluss Algorithmus zur Verfügung stehen.

Verteilen des entwickelten Evaluierungsprogrammes

Da wie im Abschnitt zuvor erwähnt das Programm auf realer Hardware getestet werden soll, um so zu aussagekräftigen Ergebnissen zu kommen, wurde eine Funktion eingebaut mit der die Testpersonen die Ergebnisse einfach durch einen Klick auf einen Sendebutton via Mail versenden können. Dies scheint auf den ersten Blick relativ einfach zu sein, doch hierbei haben sich die Tücken von den zig verschiedenen Android APIs sofort bemerkbar gemacht. Einerseits musste eine sehr alte API verwendet werden, da die neuen nicht immer rückwertskompatibel sind und andererseits mussten auch Anpassungen an verschiedene Android Versionen vorgenommen werden. Selbst innerhalb der Version 4 von Android gab es einige Unterschiede z. B. konnte mit Version 4.4 die Auswahl des Programmes mit der der Evaluierungsbericht gesendet werden sollte ohne Probleme ausgewählt werden (über einen Auswahldialog) während dies mit Version 4.1 nicht möglich war.

Als die Probleme mit den verschiedenen Android Versionen behoben waren, konnte das selbst signierte Programm an einige Testuser verteilt werden. Dabei wurde darauf Wert gelegt, soweit wie möglich einige aktuelle Smartphones zu testen. Die Ergebnisse der Auswertung werden in einem eigenen Unterpunkt in Kapitel 4 besprochen.

Auswerten der Ergebnisse

Die Evaluierungsberichte wurden via Mail von den Testbenutzern übermittelt. Zusätzlich wurden einige Geräteinformationen abgefragt. Um die Ergebnisse besser vergleichen zu können, wurde

die selbe Testsequenz ebenfalls auf einem Standard X86 PC ausgeführt. Die Ergebnisse wurden aufbereitet und sind in Kapitel 4 ersichtlich. Ein Ergebnis sollte bereits wird vorweg erwähnt werden: die Leistung eines modernen Smartphones gegenüber einem Standard X86 PC ist etwas überraschend da die Leistung deutlicher höher als erwartet ausfällt.

3.4 Hinderniserkennung

Für die Hinderniserkennung ist es notwendig, die bereits zuvor angesprochenen Funktionen, also den FAST Algorithmus sowie den Optischen Fluss, um weitere Funktionen zu erweitern. Führt man nur FAST und den Optischen Fluss aus, kann man zwar eine simple Hinderniserkennung implementieren, aber es ist damit alleine nicht möglich, absolute Distanzen zu Hindernissen zu bestimmen. Um dies jedoch zu ermöglichen, wird auf den Focus of Expansion und Time to Contact zurückgegriffen, die in den einzelnen Unterpunkten im Detail besprochen werden. Auch wird kurz auf den FAST Feature Detektoren und dessen Geschwindigkeitsvorteil gegeünber anderen Feature Detektoren eingegangen.

3.4.1 Feature Detektoren

Bisher wurde bereits der Optische Fluss besprochen, die Grundvoraussetzung aber, erstmal markante Punkte zu detektieren, wurde bisher nur am Rande diskutiert. Der Feature Detektor ist sehr wichtig, da es ohne Feature Detektor keine Features, also markante Punkte, welche die Ausgangsbasis für das Verfolgen dieser via dem Optischen Fluss darstellen, gibt. In dieser Arbeit wird ein FAST [RD05], [RD06] Feature Detektor verwendet. Dieser bietet im Vergleich zu anderen üblichen Feature Detektoren, wie dem Good Features to Track Detektor, weniger Anpassungsmöglichkeiten was die Detektion der Features angeht, hat aber den sehr großen Vorteil, dass er extrem schnell ist. Die geringe Ausführungszeit kommt natürlich einem Embedded System zu Gute, da sich dadurch wieder wichtige Millisekunden an Ausführungszeit einsparen lassen. Die Verwendung des FAST Feature Detektors in dieser Arbeit bei realen Szenen war jedoch etwas schwieriger, als man vielleicht annehmen würde. Die realen Szenen haben oft wenig Textur oder aber die Bilder haben eine nicht ausreichend gute Bildqualität um über das gesamte Bild hinweg gleichmäßig verteilt Features zu detektieren. Eine simple aber schlechte Möglichkeit wäre den Threshold des FAST Algorithmus zu verringern. Dies würde nämlich dazu führen, dass man einerseits mehr (je nach Threshold extrem viel mehr) Features detektiert, diese aber weder gut verteilt sind, noch will man zu viele Features haben. Hat man zu viele Features, erhöht das einerseits den Detektionsaufwand, andererseits müssen alle gefundenen Features in den nächsten Schritten u.a. durch den Optischen Fluss Algorithmus weiter verarbeitet werden. Eine bessere Lösung hierfür, ist der Einsatz eines Raster-basierten FAST Feature Detektors mit einem zusätzlichen Limit für die Anzahl der Features inklusive einem Mindestabstand dieser. Um den Mindestabstand zu realisieren, wird nach der ersten Detektion eine Maske um den entsprechenden Radius um die bereits gefundenen Features gelegt. Dies wird in Abbildung 3.16 anschaulich dargestellt.

Als Rastergröße wird ein 5x4 Feld gewählt, um so das 640x480 Pixel große Bild zu unterteilen. Als Limit für die Anzahl der Features werden 500 ausgewählt, sowie ein Mindestabstand von zehn Pixel zueinander. Durch diese Maßnahmen erreicht man eine gute Verteilung der Features über den gesamten Bildbereich. Ein Vergleich einzelner Feature Detektoren ist hier zu finden [13]. Das Interessante an diesem Test ist vor allem, dass nicht nur die Performance auf einem

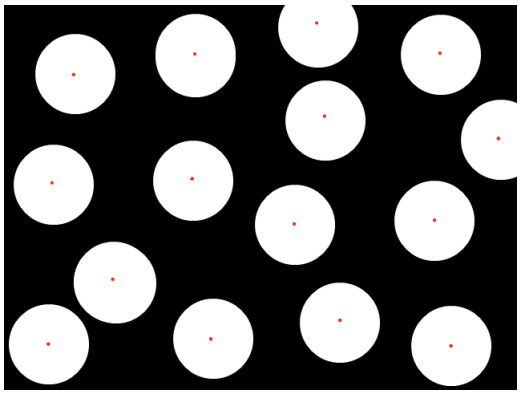


Abbildung 3.16: Maske um einen Mindestabstand zwischen den verschiedenen Features herzustellen. Die roten Punkte sind die derzeitigen Features. Die weißen Flächen sind der Bereich, in dem nicht gesucht werden darf, d.h. nur im schwarzen Bereich können neue Features detektiert werden. So kann ein Mindestabstand zwischen den Features eingehalten werden.

Standard X86 PC evaluiert wurde, sondern auch ein etwas älteres iPhone 3GS, welches in etwa der Leistung eines Gumstix Overo entspricht. Ein Vergleich der Feature Detektoren sowie deren Ausführungszeit ist in Abbildung 3.17 angeführt. Die Ergebnisse spiegeln auch die Erfahrung, die in dieser Arbeit gesammelt wurde, wieder.

3.4.2 Focus of Expansion

Der Focus of Expansion (FOE) ist der Punkt, von dem sich alle Vektoren des Optischen Flusses entfernen. Der FOE gibt außerdem die Richtung (heading) an. Bewegt man sich rückwärts, bewegen sich alle Vektoren zum FOE hin, dieser Punkt wird dann aber nicht als Focus auf Expansion sondern als Focus of Contraction (FOC)bezeichnet. Bewegt man sich nur geradeaus, ohne, dass eine Rotation berücksichtigt werden muss, also, dass die Bewegung eine reine Translation ist, wird der FOE sich in der Mitte des Bildes befinden. Würde man den einen Dense Optical Flow Algorithmus verwenden, der jeden Pixel des Bildes berechnet, würde der FOE der Punkt sein, an dem keine Bewegung stattfindet, da sich alle Vektoren des Optischen Flusses radial vom Focus of Expansion entfernen - dies ist in Abbildung 3.18, sowie in Abbildung 2.8 zu sehen.

Da das Berechnen für jeden Pixel natürlich bei Weitem zu viel Rechenaufwand darstellt, wird wie bereits in den Punkten zuvor erwähnt, nur ein Sparse Optical Flow Algorithmus eingesetzt. Der einfache Ansatz nur den Vektor zu ermitteln, der am kürzesten ist, macht somit wenig Sinn und es muss ein anderer Ansatz gewählt werden. Theoretisch würde es reichen, wenn man nur zwei Vektoren des Optischen Flusses bestimmt und dann durch diese eine Linie zieht. Der Punkt an dem sie sich schneiden wäre der Focus of Expansion. In der Realität ist dies so nicht möglich, da die Bilder und der berechnete Optische Fluss nicht 100% exakt sind. In dieser Arbeit wird als Ansatz die Methode der kleinsten Quadrate verwendet um den FOE zu berechnen [TGS91].

$$\begin{aligned} FOE &= (A^T A)^{-1} A^T \vec{B} \\ &= \begin{bmatrix} \sum a_{i0} b_i \sum a_{j^2_1} & - & \sum a_{i1} b_i \sum a_{j0} a_{j1} \\ \sum a_{i0} b_i \sum a_{j0} a_{j1} & + & \sum a_{i1} b_i \sum a^2_{j0} \end{bmatrix} \frac{1}{\sum a_{j^2_0} a_{j^2_1} - (\sum a_{i0} a_{i1})^2} \end{aligned} \tag{3.1}$$

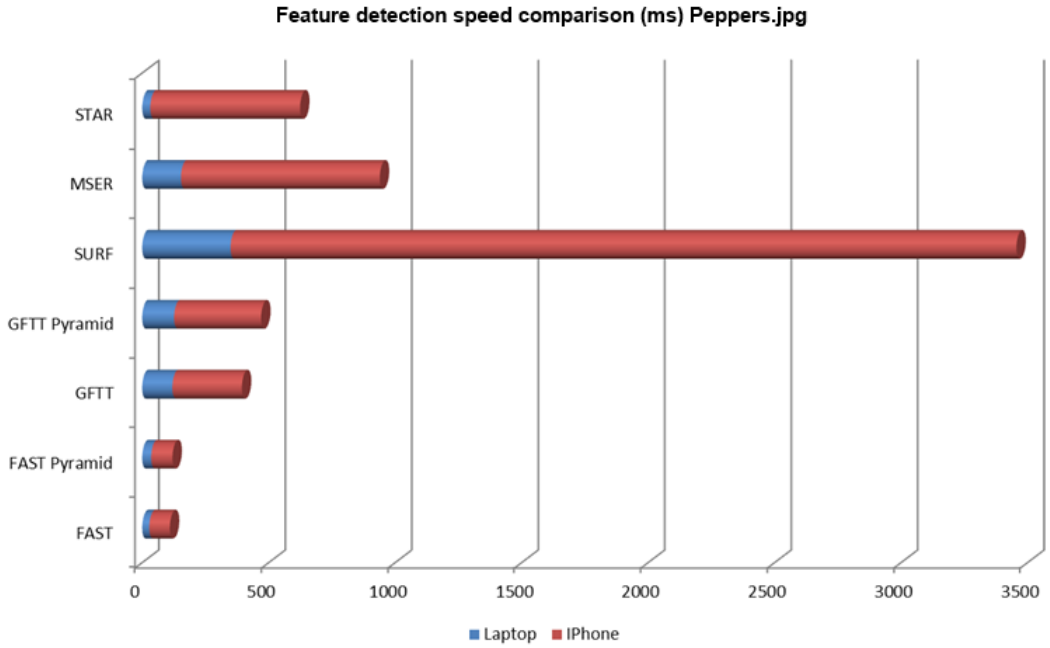


Abbildung 3.17: Feature Detektor Performance Vergleich. Dabei ist deutlich zu sehen, dass der FAST Feature Detektor sowohl am Laptop (blau) und am iPhone (rot) deutlich schneller als vergleichbare Feature Dektoren ist. Gerade der Vergleich mit dem iPhone ist hierbei interessant, da dieses in etwa die gleiche Leistung zur Verfügung stellt wie ein Gumstix Overo Fire. Abbildung von Eugene Khvedchenya.

$$A = \begin{bmatrix} a_{00} & a_{01} \\ \dots & \dots \\ a_{n0} & a_{n1} \end{bmatrix} \qquad \overrightarrow{B} = \begin{bmatrix} b_0 \\ \dots \\ b_n \end{bmatrix} \tag{3.2}$$

Hierbei entspricht

$$a_{i0} = v$$

$$a_{i1} = -u$$

sowie

$$b_i = xv - yu$$

Wobei u und v die Werte der Optischen Fluss Vektoren sind. Durch diese Formel ist es nun möglich, den Focus of Expansion zu bestimmen. Der FOE kann jedoch nur berechnet werden wenn es eine Translationsbewegung gibt. Ohne Translationsbewegung befindet sich der berechnete Punkt außerhalb der Bildebene. Dies stellt jedoch kein Problem dar. Man kann es sich wie folgt vorstellen: Dreht sich die Drohne im Kreis, oder ändert sie ihre Höhe, wird auch die Entfernung zu den Objekten nicht verändert da ja keine Vorwärtsbewegung also Translation stattfindet. Der Nachteil, der sich aus diesem Umstand ergibt, wurde bereits in Kapitel 2 besprochen. Ohne Translation ist es nämlich, anders als bei Stereokamera Systemen, nicht möglich, Tiefeninformationen zu gewinnen. Dieser Zusammenhang wird im Unterpunkt Rotationskompensation formelmäßig erläutert.

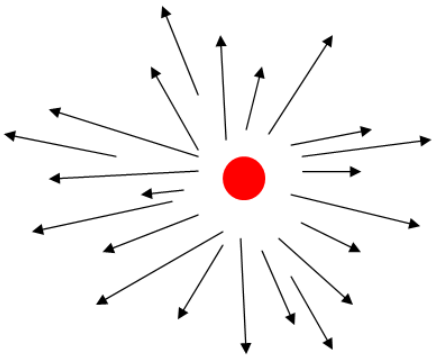


Abbildung 3.18: Der Focus of Expansion entspricht dem roten Punkt. Wie man sehen kann, entfernen sich alle Vektoren des Optischen Flusses radial vom FOE.

3.4.3 Time to Contact

Nachdem man zuvor den Focus of Expansion berechnet hat, lässt sich nun die Time to Contact (TTC), oder auch als time to impact bzw. time to collision bezeichnet, berechnen. Bei dem Ansatz der in dieser Arbeit umgesetzt wird, geht man davon aus, dass man die Länge der Vektoren des Optischen Flusses (y) sowie den Fokus of Expansion bereits zuvor berechnet hat. Nachdem dies geschehen ist, berechnet man über trigonometrische Beziehungen die Entfernung vom FoE zur Spitze des Optischen Fluss Vektors, welcher hier als x bezeichnet wird. Nachdem man die Entfernung hat, kann man die Länge des Optischen Fluss Vektors durch die Entfernung dividieren und für τ, welches als Zeit bis zum Kontakt bezeichnet wird, einsetzen.

$$\tau = \frac{x}{y}$$

Hierbei stellt τ jedoch nur eine relative Angabe der Entfernung dar. Will man sich die absolute Entfernung berechnen, muss die Geschwindigkeit der Drohne bekannt sein und als konstant angenommen werden. Die Geschwindigkeit kann z. B. über visuelle Odometry oder über die GPS Informationen, welche ein im Smartphone verbauter GPS Empfänger liefern könnte, berechnet werden. Die Bestimmung über GPS funktioniert natürlich nur, wenn sich die Drohne in freier Umgebung befindet. Diese Anforderung erfüllt die ICT Drohne. Hat man also die Geschwindigkeit bestimmt, kann man nun durch Umformen der bekannten Formel

$$Geschwindigkeit = \frac{Weg}{Zeit}$$

die Entfernung bestimmen.

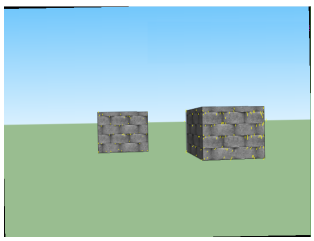
$$Weg = Geschwindigkeit * Zeit$$

Wobei hier der Weg dem Abstand zum Objekt entspricht. Die Berechnung muss für jeden Vektor erneut durchgeführt werden. Da es gerade bei Szenen in realer Umgebung immer wieder Ausreißer geben wird und diese nicht immer 100% eliminiert werden können, ist es notwendig, eine Grenze

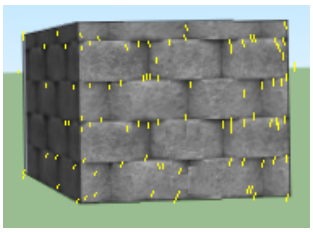
einzuführen, ab welcher die erhaltenen Werte für die Hinderniserkennung berücksichtigt werden. Ein Ansatz wäre z. B., dass man das Bild in Blöcke unterteilt und immer das arithmetische Mittel oder den Median für τ pro Block berechnet. Dadurch würden Ausreißer einen geringeren Einfluss haben.

3.4.4 Rotationskompensation

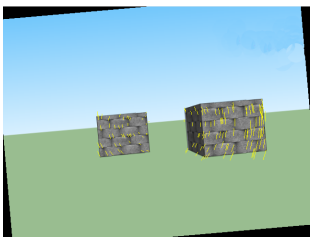
Die Rotationskompensation ist einer der wichtigsten Schritte bei der Hinderniserkennung wenn man den time to contact Ansatz verwendet. Dies hat den Grund, dass die Vektoren des Optischen Fluss sich im Prinzip aus zwei Komponenten zusammensetzen, dem Translations- sowie dem Rotationsanteil. Das Problem an der Rotation und warum man diese aus den Optischen Fluss Vektoren herausrechnet ist, dass sie keine Tiefeninformationen enthalten. Dies wird auch deutlich, wenn man sich die Formel 3.5 anschaut. Würde man keine Rotationskompensation durchführen, wäre die Entfernung zu den Objekten, je nach Stärke der Rotation, extrem verfälscht. In Abbildung 3.19 sieht man eine Rotation um ein, fünf und zehn Grad. Die abgebildete Bewegung der synthetischen Szene besteht aus einer reinen Rotation ohne Translationsbewegung. Die Vektoren des Optischen Flusses sind aber trotzdem, je nachdem wie viel Grad die Rotation beträgt, durchaus lang. Dieser Umstand bewirkt nun, dass das Objekt näher erscheint, als es wirklich ist. Bei einer reinen Rotation, wie es in der Abbildung veranschaulicht ist, würde die Entfernung der Objekte zwischen den zwei Bildern immer gleich bleiben.



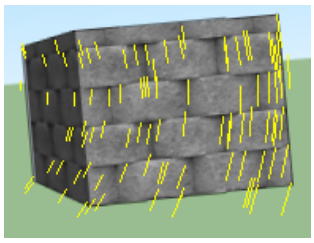
(a) Rotation um 1 Grad



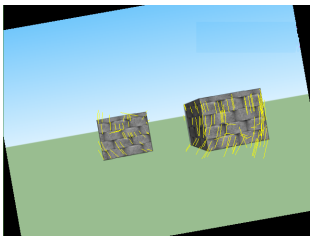
(b) Bildausschnitt Rotation 1 Grad



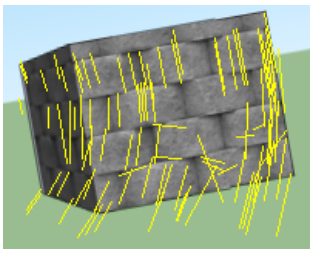
(c) Rotation um 5 Grad



(d) Bildausschnitt Rotation 5 Grad



(e) Rotation um 10 Grad



(f) Bildausschnitt Rotation 10 Grad

Abbildung 3.19: Eine reine Rotationsbewegung um 1, 5 sowie 10 Grad. Die gelb eingezeichneten Linien sind die Optischen Fluss Vektoren. Um die Vektoren des Optischen Flusses besser sichtbar zu machen, wurde jeweils ein Bild der gesamten Szene, sowie ein Bildausschnitt welcher den rechten Würfel der Szene zeigt, dargestellt

Um die Rotationskompensation durchzuführen, kann auf zwei Ansätze zurückgegriffen werden. Die Vorteile der beiden Ansätze, sowie deren Schwierigkeiten bei der Implementierung werden ebenfalls diskutiert. Die Ergebnisse der Rotationskompensation, sind wie in dieser Arbeit üblich, in Kapitel 4 unter dem entsprechenden Unterpunkt aufgeführt.

Ansatz 1: Verwendung eines Gyroskops

In einigen Arbeiten bzw. Projekten, in welchen ebenfalls Rotationskompensation notwendig ist, wird der Ansatz, die Rotation über das Gyroskop herauszurechnen, gewählt. Das Gyroskop wird hierbei jedoch selten allein verwendet. Dies hat den Grund, dass sich die Ergebnisse der Gyroskopmessung immer weiter vom realen Wert wegbewegen, der sogenannte Gyroskopdrift [PH12, S. 23 ff.]. Diesen kann man kompensieren, indem man wie Peter Hanger einen Komplementärfilter verwendet, um so in Zusammenarbeit mit Beschleunigungssensoren sowie Magnetfeldsensoren, die Roll-, Nick und Gierachse zu korrigieren und somit den Drift soweit als wie möglich zu verringern. Statt einem Komplementärfilter besteht auch die Möglichkeit die Sensorfusion über einen Kalmanfilter [WB95] (oder auch erweiterten Kalmanfilter) zu realisieren.

Ansatz 2: pose estimation

Eine andere Möglichkeit, die nicht direkt von der Hardware, also einem Magnetfeldsensor, Gyroskop oder auch Beschleunigungssensor abhängig ist, ist die Bestimmung der Lage rein über die von der Kamera gewonnen Bilddaten. Diese Methode wird pose estimation genannt. Die Pose Berechnung ist von der Rechenzeit etwas aufwendiger als ein Komplementärfilter, dafür entfällt jedoch die Synchronisation der Sensordaten, die man von Hardwaresensoren erhalten hat, mit den Bilddaten. Wie die pose estimation funktioniert und welche Schwierigkeiten dabei auftreten, wird nachfolgend diskutiert. Bei der pose estimation, wie sie in dieser Arbeit umgesetzt ist, werden immer zwei aufeinanderfolgende Bilder t_n und t_{n+1} verglichen und der Unterschied der Lage dieser zwei Bilder berechnet.

Die Berechnung der Pose erfordert einige Schritte. Hier wird jedoch nur kurz auf die wichtigsten Punkte eingegangen, da dies sonst den Umfang dieser Arbeit sprengen würde. Für Details sei auf [HZ04] Kapitel 9 verwiesen.

- Erkennung markanter Punkte durch FAST
- Verfolgen der markanten Punkte mittels Optischen Fluss
- Berechnen der Fundamentalmatrix
- Berechnen der Essentiellen Matrix
- Berechnen der Kameramatrix P
- Triangulation der Punkte mittels der in der Kameramatrix P enthaltenen Rotations- und Translationsparameter

Den FAST Algorithmus und das Verfolgen von markanten Punkten von Bild t_n zu Bild t_{n+1} wurde bereits zuvor in dieser Arbeit angesprochen. Aus diesem Grund wird nur auf die restlichen Punkte näher eingegangen.

Berechnen der Fundamentalmatrix

Die Fundamentalmatrix ist eine 3x3 Matrix und wird zwischen den zwei Bildern berechnet. Einfach ausgedrückt, stellt sie die Beziehung zwischen den korrespondierenden Punkten in den zwei Bildern da. Berechnet man die Fundamentalmatrix mittels der von OpenCV zur Verfügung gestellten Funktion *findFundamentalMat*, kann man auch wählen, mittels welchem Algorithmus diese Berechnung durchgeführt wird. Zur Auswahl steht der 7 Point, 8 Point, RANSAC [FB81] sowie LMEDS Algorithmus. In dieser Arbeit wird der RANSAC (Random sample consensus) Algorithmus verwendet. Für die Berechnung der Fundamentalmatrix ist es erforderlich, dass mindestens acht Punktpaare vorhanden sind, ansonsten kann die Berechnung nicht durchgeführt werden. Der RANSAC Algorithmus wird im Folgenden kurz erklärt. Beim RANSAC wird zuerst ein Modell aufgestellt bzw. berechnet. Anschließend werden die Modellparameter berechnet. Danach wird mittels dem zuvor festgelegten Grenzwert eine Teilmenge berechnet und (in dieser Arbeit hat sich für den OpenCV Parameter ein Grenzwert von < 1 und als Vertrauenswert 99% als sinnvoll herausgestellt) festgestellt, welche Werte innerhalb des Grenzwertes (Inliner) und welche sich außerhalb (Outliner) befinden. Dieser Vorgang wird mehrmals durchgeführt, bis sich eine Teilmenge herauskristallisiert, die die meiste Anzahl an Punkten enthält. In Abbildung 3.20 wird ein beispielhaftes Resultat einer Berechnung sowie die Grenzwerte bzw. Schranken bildlich dargestellt.

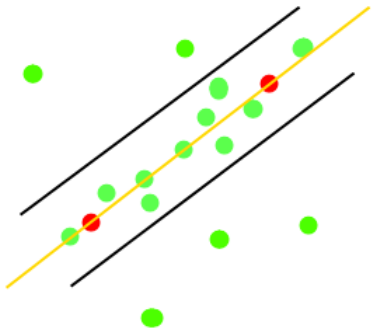


Abbildung 3.20: RANSAC, rot zwei zufällig gewählte Punkte, schwarze Linien die Schranken. Die Punkte außerhalb der schwarzen Linien sind die Outliner, die Punkte innerhalb der Schranken die Inliner

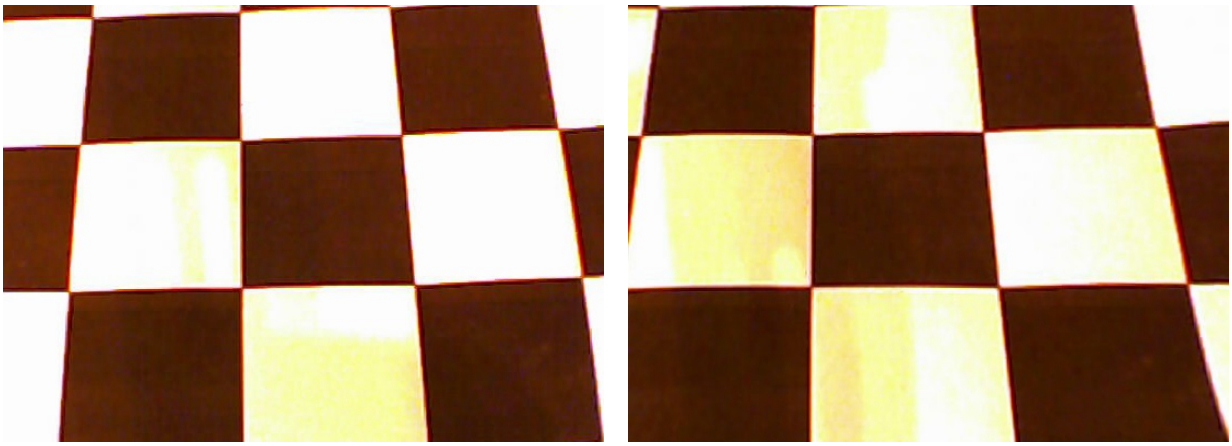
Dadurch, dass man für die Fundamentalmatrix-Berechnung ebenfalls den RANSAC Algorithmus verwendet, hat man den Vorteil, dass man gleichzeitig Ausreißer eliminiert, was ein positiver Nebeneffekt der Fundamentalmatrix-Berechnung ist. Es werden nicht alle, aber zumindest ein Großteil der Ausreißer erkannt. Um die verbleibenden Ausreißer noch sinnvoll zu erkennen und aus den weiteren Berechnungen auszuschließen, sind weitere Checks notwendig.

Berechnen der Essentiellen Matrix

Die Essentielle Matrix berechnet sich folgendermaßen:

$$E = K^T * F * K \tag{3.3}$$

Wobei hier K der Kamera-Kalibrationsmatrix und F der Fundamentalmatrix entspricht. Die Kamera-Kalibrierungsmatrix in die Berechnung einfließen zu lassen ist ein wichtiger Schritt, da dadurch die von der Optik verursachten Verzerrungen minimiert werden können. Dies wurde auch bereits in Kapitel 2 angesprochen. Die Kamera-Kalibrierungsmatrix kann entweder via einer Matlab Toolbox oder direkt über OpenCV bestimmt werden. In dieser Arbeit wird die OpenCV Variante gewählt, da bereits mit OpenCV gearbeitet wird. Dazu ist es notwendig, dass ein Schachbrettmuster aus verschiedenen Perspektiven aufgenommen wird. Anschließend können die errechneten Werte zur Korrektur in ein File abgespeichert werden und im eigentlichen Programm zur Berechnung der Essentiellen Matrix verwendet werden. In Abbildung 3.21 (a-b)



(a) Schachbrettmuster kalibriert (b) Schachbrettmuster unkalibriert

Abbildung 3.21: Schachbrettmuster vor und nach der Kalibration. Besonders zu beachten ist hierbei die rechte, untere Ecke des Schachbrettmusters. Hier ist die Verzerrung am deutlichsten zu erkennen. Die braunen Flecken in den weißen Kästchen des Schachbrettmusters sind ebenfalls der schlechten Qualität der Webcam und der Beleuchtung geschuldet.

sieht man einmal ein original Kamera Bild des Schachbrettmusters und das korrigierte. Wie man anhand der Bilder erkennen kann, ist die Verzerrung an den Rändern deutlich sichtbar. Umso größer nun die Entfernung zu einem Objekt ist, umso größer wird auch der Fehler. Man erkennt also deutlich wie wichtig es ist die Kamera zu kalibrieren. Die Aufnahme des Schachbrettmusters wurde mit einer 1.3 Megapixel Philips SPC230NC Webcam durchgeführt.

Berechnen der Kameramatrix P

Während das Berechnen der Fundamentalmatrix und der Essentiallen Matrix, nachdem man bereits vorab die Kalibrierungsmatrix bestimmt hat, relativ einfach ist, ist das Berechnen der Kameramatrix bereits aufwendiger. Die Essentiale Matrix muss mittels einer Singulärwertzerlegung bestimmt werden. Dabei erhält man eine 3x4 Kameramatrix P die die Pose angibt.

$$P = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \end{bmatrix} \tag{3.4}$$

Dabei ist zu beachten, dass die Rotationswerte und die Translationswerte jeweils negativ und positiv sein können. Dies bedeutet nun konkret, dass man allein mit der Kameramatrix P vier

verschiedene Lösungen bekommt, wie sich die zwei Kameras bzw. Bilder t_n und t_{n+1} zueinander verhalten. Dieser Zusammenhang wird in Abbildung 3.22 (a-d) verdeutlicht. Aus diesem Grund ist noch ein weiterer Berechnungsschritt notwendig, um die richtige der vier Lösungen bestimmen zu können.

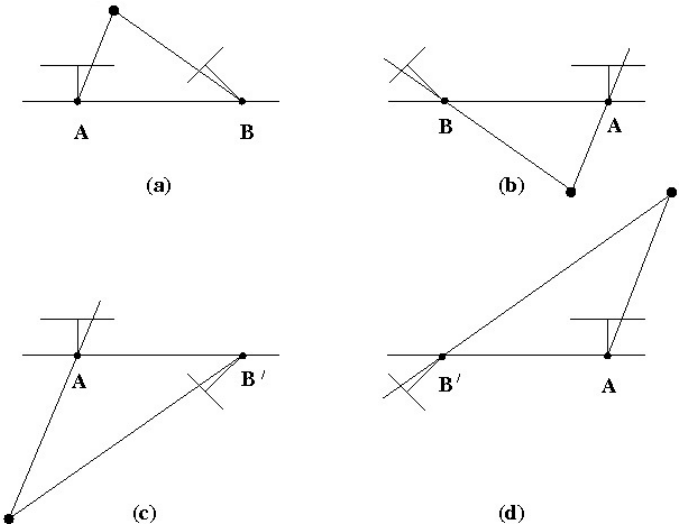


Abbildung 3.22: Darstellung der vier Möglichkeiten wie die zwei Kameras angeordnet sein könnten. Nur eine der vier Lösungen entspricht der korrekten Abbildung. [HZ00]

Triangulation der Punkte

Die Triangulation der Punkte wird verwendet um den Zusammenhang der Bilder und somit eine eindeutige der vier Lösungsvarianten zu bestimmen. Durch die Triangulation kann bestimmt werden welche Kameras die Features der Bilder vor sich haben, also ob die Z Ebene (Tiefe) korrekt ist. Ist dies der Fall, ist die Lösung die Richtige und die Werte für die Rotation können verwendet und wenn nötig in Eulerwinkel oder Quaternion umgerechnet werden. Bei der Berechnung der Triangulation würde es theoretisch reichen wenn man dies nur für einen Punkt durchführt. Es ist jedoch zu empfehlen, eine größere Anzahl zu verwenden, da man sich nicht 100% sicher sein kann, dass gerade der ausgewählte Punkt kein Ausreißer ist und somit das Ergebnis verfälschen würde.

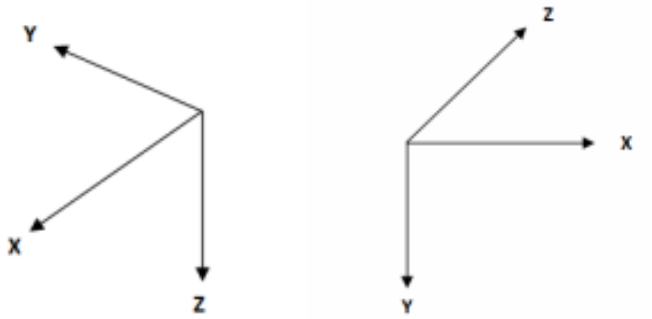
Da die Vektoren des Optischen Flusses sowie die Rotation berechnet und nun bekannt sind, kann mit der Rotationskompensation fortgeschritten werden. Die Vektoren des Optischen Flusses sind linear teilbar, dies macht man sich bei der Rotationskompensation zu nutzen. In [Kan88] wird detailliert darauf eingegangen, wie man diese Formel herleiten kann und welche mathematischen Zusammenhänge dazwischen bestehen. Da der Umfang dieser Arbeit im Rahmen bleiben soll, wird hier nur auf die wesentlichen Teile eingegangen, die schlussendlich für die Rotationskompensation notwendig sind. In der Literatur werden verschiedene Schreibweisen für die selbe Formel, teilweise andere Buchstaben, teilweise Matrix Form, teilweise bereits die Fokallänge berücksichtigt etc. Hier wird statt der Schreibweise wie in [Kan88] eine verwendet, wie sie auch in [LW05] Verwendung findet, da bei dieser der Zusammenhang zwischen den Translationen T sowie den Rotationen ω besser ersichtlich ist und dies so für den interessierten Leser dieser Arbeit auch einfacher nachzuvollziehen sein sollte.

$$v_x = \frac{T_z x - T_x f}{Z} - \omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f} \tag{3.5}$$

$$v_y = \frac{T_z y - T_y f}{Z} - \omega_x f + \omega_z x + \frac{\omega_y xy}{f} - \frac{\omega_x y^2}{f} \tag{3.6}$$

Die beiden Formeln 3.5 und 3.6 sind jeweils für die x bzw. y Komponenten relevant. Wobei hier v den Optischen Fluss Vektor darstellt, T den Translationsvektor, f die Fokallänge, Z die Tiefe und ω die Rotation um die einzelnen Achsen. Wenn man sich die Formeln genau ansieht, wird deutlich, dass diese für x und y im Prinzip nur vertauschte Werte haben. Das interessantere ist jedoch, dass, wie man bereits dank dieser Schreibweise deutlich sieht, zwei Teile existieren. Einmal der Translationsteil, welcher auch die Z Komponente beinhaltet und einmal der Rotationsteil, ohne einer Z Komponente. Dadurch, dass der Rotationsanteil keinen Beitrag zur Tiefe Z darstellt, jedoch durch die lineare Abhängigkeit in den Vektoren des Optischen Flusses vorkommt, muss der Rotationsanteil entfernt werden. Dadurch wird gewährleistet, dass der Abstand zu einem Objekt korrekt berechnet werden kann.

Bei der Rotationskompensation muss besonders darauf geachtet werden, dass die Kamera und die daraus berechneten Winkel mit denen der Drohnennsteuerung übereinstimmen. Peter Hanger [PH12, S. 4] hat als Konvention für die Koordinaten das sogenannte NED-System (North-East-Down) verwendet, während die Achsen bei den Berechnungen mit OpenCV nicht dem NED-System entsprechen. Dadurch ist eine Umrechnung, sowie die gemeinsame Verwendung eines Systems unumgänglich. In Abbildung 3.23 (a-b) ist das von Peter Hanger verwendete NED System, sowie die Konvention, die OpenCV verwendet, ersichtlich.



(a) NED-System (b) OpenCV Koordinaten System

Abbildung 3.23: NED-System vs OpenCV Koordinaten System. Die Verwendungen eines einheitlichen Koordinaten Systems ist unbedingt erforderlich da ansonsten die Rotationskompensation nicht korrekt funktioniert da die Drehung auf einer falschen Achse entfernt wird.

3.4.5 Gesamtablauf der Hinderniserkennung

Die Hinderniserkennung umfasst mehrere Schritte, die notwendig sind, um die Rotations zu bestimmen und diese zu entfernen um dann anschließend den Focus of Expansion sowie die Time

to Contact zu berechnen. Auch muss das Bild vorab mit den von der Kamera Kalibrierung gespeicherten Parametern korrigiert werden. Der gesamte Ablauf mit den einzelnen Schritten wird noch einmal übersichtlich in Abbildung 3.24 dargestellt.

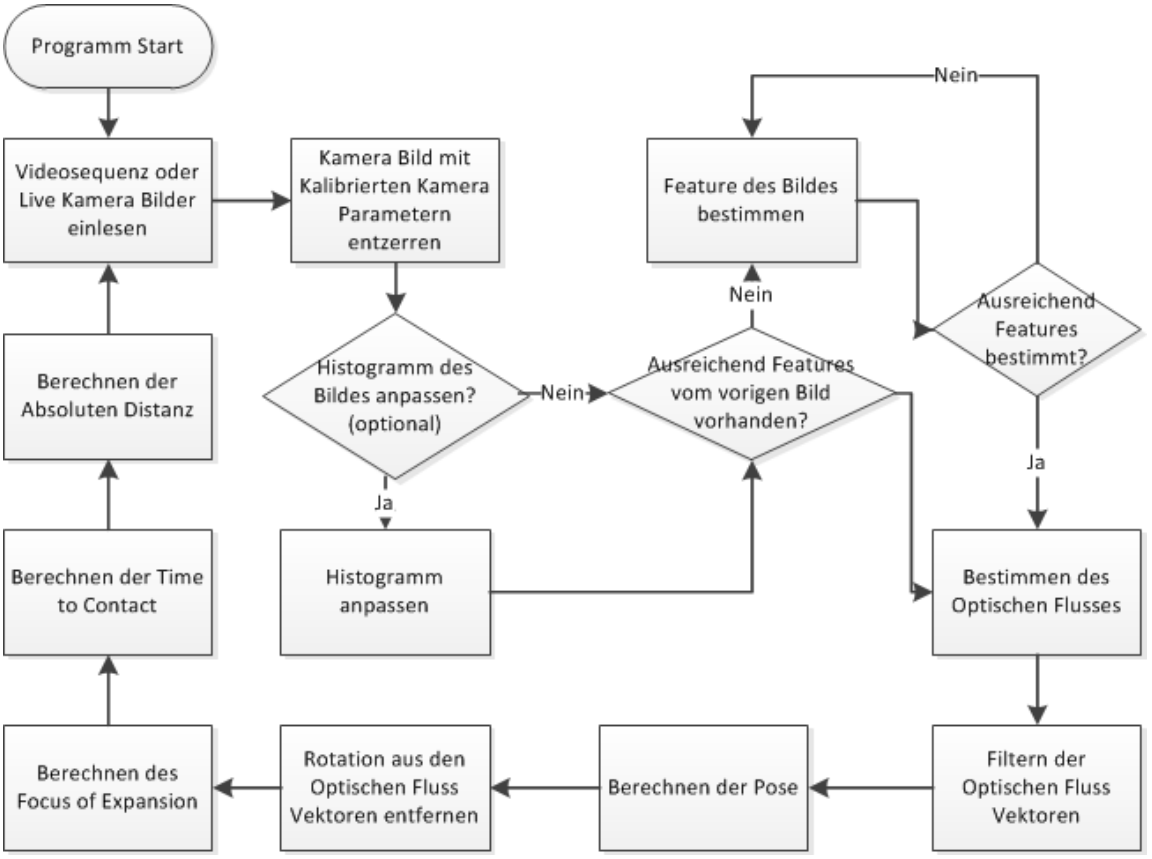


Abbildung 3.24: Schematische Übersicht über den Ablauf der Hinderniserkennung. Das Bestimmen des Histogramms ist optional und kann verwendet werden, um die Graustufenbilder, die für die Berechnung des Optischen Flusses benötigt werden, zu verbessern.

Wie man anhand der Grafik erkennen kann, ist die Bestimmung der Features wichtig. Das Repertoire an Features muss immer wieder neu befüllt werden sobald alte Features verloren gehen. Dies kann entweder dadurch passieren, dass sich der Bildausschnitt in dem sich die Features befunden haben nicht mehr im nachfolgenden Bild vorhanden ist, oder aber auch durch falsches Tracking und anschließendem Herausfiltern der Features. Eine Anzahl von ca. 300-500 Features hat sich als ausreichend erwiesen.

Optional ist es auch möglich, das Histogramm der Bilder erst anzupassen, nachdem die Bilder mit den kalibrierten Kameraparametern angepasst wurden. Die Verwendung des Histogramms hat einerseits den Vorteil, dass die Bilder, welche für den Optischen Fluss in Graustufen sind, im eigentlichen Sinne nicht verändern werden. Andererseits erfolgt dadurch eine bessere Verteilung der Graustufen über das gesamte Bild. Der Effekt der Anpassung des Histogramms ist in Abbildung 3.25 zu sehen. Da dies je nach Umgebungslicht und verwendeter Kamera nicht immer benötigt wird, ist dies optional zu verwenden.

(a) Originalbild in Graustufen (b) Bild mit angepasstem Histogramm

Abbildung 3.25: Anpassung des Histogramms des Graustufenbildes. Bei Bild (a) ist das Original zu sehen bzw. sieht man fast nur eine schwarze Fläche. In Abbildung (b) sieht man das Bild nachdem das Histogramm angepasst wurde. Statt einer schwarzen Fläche sind jetzt deutlich mehr Details eines Hauses zu erkennen. Dies ermöglicht auch ein besseres Finden sowie Tracken der Features.

4 Ergebnisse

In den nachfolgenden Unterpunkten werden die Ergebnisse des Vergleichs zwischen FastCV und OpenCV zwischen verschiedenen Smartphones, Tablets, Overo Fire und einem X86 PC sowie die der Umsetzung der Hinderniserkennung diskutiert und beurteilt. Außerdem wird die Performance eines Smartphones als Hardware, welche aufgrund der mangelden Performance der Gumstix Plattform notwendig ist, für die Drohnenplattform analysiert. Im Vorfeld muss erwähnt werden, dass die Hinderniserkennung nicht wie gewünscht umgesetzt werden konnte, da während der Erstellung der Hinderniserkennung Probleme aufgetreten sind. Es konnten außerdem neue Erkenntnisse bezüglich der Schwachstellen der Hinderniserkennung mittels Mono-Kamera-System gewonnen werden. Diese Probleme und Erkenntnisse führen dazu, dass die Hinderniserkennung bzw. die Bestimmung der Entfernung zu einem Hindernis nicht genau genug bestimmt werden kann. Die Details dazu werden in den entsprechenden Unterkapiteln näher erläutert.

4.1 FastCV und OpenCV - Performanceanalyse

Um einschätzen zu können, wie gut sich ein modernes Smartphone für den Einsatz als Hauptcomputer der Drohne eignet, wurde ein Android Programm entwickelt welches die Performance für die Feature Detektion sowie den Optischen Fluss Algorithmus misst. Die Testsequenz besteht aus mehreren Abschnitten, da OpenCV und FastCV sowie die verschiedenen Performance Modi der Snapdragon Chips ebenfalls bewertet werden sollen. Die verwendeten Modi der FastCV 1.3 API, auf welche für die Implementierung der Testapp zurückgegriffen wurde, sind in Tabelle 4.1 kurz beschrieben.

FASTCV_OP_LOW_POWER	hierbei wird die QDSP Implementierung verwendet wenn die Geschwindigkeit des QDSP nicht 3x langsamer ist als die der CPU
FASTCV_OP_PERFORMANCE	die schnellste Implementierung wird verwendet
FASTCV_OP_CPU_OFFLOAD	die QDSP Implementierung wird verwendet sofern diese verfügbar ist, ansonsten wird entweder auf die GPU oder CPU Implementierung zurückgegriffen

Tabelle 4.1: Beschreibung der verschiedenen verfügbaren Modi, welche die FastCV API anbietet [Inc13]. Für die Implementierung der Testapp wurde jede Testsequenz mit jedem Performance Modus ausgeführt, um Unterschiede in der Ausführungszeit der Algorithmen festzustellen.

Da die Modi nur auf einem Smartphone mit Snapdragon Chip wirksam sind und nicht auf einem PC mit X86 Chip oder einem Smartphone mit einem nicht Qualcomm Chip wie z.B. dem Nvidia Tegra oder Samsung Exynos, werden die Ergebnisse der einzelnen Modi zusammengefasst. Dies wird deshalb gemacht, da die Ausführungszeiten für die einzelnen Modi zwar erfasst wurden und minimal unterschiedliche Ausführungszeiten aufweisen, welche allerdings nicht aussagekräftig sind. Die unterschiedlichen Ausführungszeiten rühren wahrscheinlich daher, dass Daten von einer vorigen Ausführung bereits im RAM oder Cache bzw. einer Pipeline vorhanden sind und so auf diese minimal schneller zugegriffen werden kann. Für diese Annahme spricht, dass die Unterschiede in der Ausführungszeit hierbei maximal 3.09% und minimal 2.4% betragen.

Eine kurze Übersicht, über die bei der Testapp verwendeten Smartphones gibt Tabelle 4.2. Bei der Testapp wurden sechs Smartphones, ein Tablet, der Gumstix Overo Fire, sowie ein X86 PC mit Core 2 Duo verwendet. In der Tabelle werden außerdem der verwendete Chip sowie die Taktung, die CPU Kerne und der Arbeitsspeicher aufgelistet.

Hersteller	Model	Chipsatz	Taktfrequenz	CPU Kerne	RAM
Samsung	Galaxy S3 (S. S3)	Samsung Exynos 4412	1.40 GHz	4	1 GB
Samsung	Galaxy S4 (S. S4)	Snapdragon 600	1.90 GHz	4	2 GB
Samsung	Galaxy Tab 3 (S. Tab 3)	Samsung Exynos 4	1.5 GHz	2	1 GB
LG	Nexus 4 (LG N4)	Snapdragon S4 Pro	1.50 GHz	4	2 GB
LG	Nexus 5 (LG N5)	Snapdragon 800	2.26 GHz	4	2 GB
LG	G2 (LG G2)	Snapdragon 800	2.26 GHz	4	2 GB
Motorola	G (M. G)	Snapdragon 400	1.2 GHz	4	1 GB
Gumstix	Overo Fire	Omap 3530	720 Mhz	1	256 MB
HP	8510p	Core 2 Duo	2.4 Ghz	2	4 GB

Tabelle 4.2: Übersicht der zum Test verwendeten Geräte. Auffällig ist die relativ identische Ausstattung der eher im High End Bereich angesiedelten Geräte, wie z.B. dem Nexus 5 und dem LG G2. Weiters sieht man, dass kein Gerät mit Tegra Chip verwendet wurde, dies hat den Grund, dass diese in Smartphones zum Test nicht zur Verfügung stand.

Alle getesteten Android Geräte hatten bereits die Version 4.1.2 oder neuere Software installiert. Die Android Version kann durchaus auch Unterschiede bei den Ergebnissen hervorrufen, da neuere Versionen oft eine bessere Performance bieten. Aufgrund der Android Update Politik (bzw. jener der Hersteller), bei der ältere Geräte keine Updates bekommen, war es nicht möglich, dass alle zur Verfügung stehenden Geräte dieselbe Android Version verwenden. Zusätzlich dazu wäre es für den Testbenutzer nicht zumutbar gewesen, sein Smartphone komplett neu zu installieren und einzurichten. Bedingt durch diese zwei Umstände, sowie möglicherweise im Hintergrund laufender Systemdienste, die mehr oder weniger viel Ressourcen verbrauchen, kann es sein, dass Geräte die eine idente Ausstattung aufweisen eine unterschiedliche Performance liefern. Dieses Verhalten wurde auch bei den Tests zwischen dem Nexus 5 und dem LG G2 beobachtet. Die genauen Ergebnisse der Performance Evaluierung zwischen den einzelnen Smartphones, dem Gumstix Overo Fire und dem PC sind in Tabelle 4.3 dargestellt. Die Messungen der einzelnen Modi bei nicht unterstützten Geräten wurde hierbei, wie oben bereits erwähnt, zusammengefasst.

Um die Ausführungszeiten zu diskutieren, werden zuerst die zwei mobilen Geräte ohne Qualcomm Chip, das Samsung Galaxy S3 und das Samsung Galaxy TAB betrachtet. Dabei fällt auf, dass beim Galaxy S3 der Unterschied zwischen OpenCV und FastCV vernachlässigt werden kann,

	Mode	S. S3	S. S4	LG N5	LG N4	LG G2	M. G	S. Tab 3	Overo	X86
OpenCV	Video	6.087	5.391	3.574	6.116	3.895	7.188	6.003	19.730	6.267
	2 Frames	0.041	0.040	0.027	0.050	0.049	0.052	0.044	0.270	0.038
FastCV V.	P.	5.976	3.428	3.618	5.197	2.714	6.619	5.440	-	4.902
	L. P.		4.036	3.224	5.108	2.737	6.711			
	CPU O.		3.520	3.101	4.853	3.436	6.563			
FastCV 2 F.	P.		0.023	0.019	0.044	0.029	0.065			
	L. P.	0.057	0.023	0.019	0.056	0.029	0.066	0.054	-	0.017
	CPU O.		0.023	0.020	0.037	0.029	0.065			

Tabelle 4.3: Ausführungszeiten der einzelnen Smartphone Modelle [1] in Sekunden. Dabei ist zu unterscheiden zwischen OpenCV und FastCV, sowie den einzelnen Performance Modi von FastCV. Auch sind die zwei verschiedenen Testsequenzen separat aufgelistet. Die Videotestsequenz, in der Tabelle als Video bezeichnet, sowie die Testsequenz, in der Tabelle als 2 Frames bezeichnet, welche nur 2 Frames besitzt.

da dieser höchstwahrscheinlich auf Messungenauigkeiten bzw. die bereits zuvor angesprochen Ursachen zurückzuführen ist. Beim Galaxy Tab 3 hingegen zeigt sich bei der Ausführung der Videosequenz ein leichter Vorteil bei der Verwendung von FastCV, während die Sequenz von nur zwei Bildern in OpenCV eine kürzere Ausführungszeit bietet. Die ähnlichen Ergebnisse, vor allem beim Galaxy S3 könnten daran liegen, dass sowohl OpenCV wie auch FastCV auf die NEON Optimierung zurückgreift. Weshalb die Ausführung der 2 Frame Videosequenz durch FastCV langsamer ist, als die von OpenCV könnte womöglich an einer unterschiedlichen Implementierung des Lucas & Kanade Optischen Fluss Algorithmus liegen. Die Videosequenz startet mit etwa 500 markanten Punkten die mittels Optischen Fluss verfolgt werden. Über die Dauer der Videosequenz gehen einige dieser markanten Punkte verloren bzw. werden herausgefiltert, wodurch zum Schluss rund 90 von den 500 markanten Punkten übrig bleiben. Durch die Reduktion der zu trackenden Punkte, nimmt auch die Ausführungszeit des Algorithmus ab. Es wäre also denkbar, dass z.B. ab einer gewissen Anzahl an zu verfolgenden Features, OpenCV bzw. FastCV effektivere bzw. weniger effektive den Algorithmus verwenden. Bei der 2. Testsequenz mit 2 Frames gehen keine Punkte verloren.

Zur weiteren Betrachtung der Ergebnisse, werden die Geräte mit Qualcomm Chip verglichen. Beim Galaxy S4, dem Nexus 4 und 5, sowie dem Motorola G ist zu beobachten, dass der CPU OFFLOAD Modus die kürzeste Ausführungszeit mit sich bringt, während das LG G2 in diesem Modus deutlich langsamer ist, als in den anderen 2 Modi. Auch fällt auf, dass das LG G2, wenn man vom besten Wert ausgeht, deutlich schneller ist als das, von der Hardware identische, Nexus 5. Beide verwenden die gleichen Android Kernel, jedoch wurde vom Tester des LG G2 erwähnt, dass dieser ein Custom ROM, also eine alternative Firmware und nicht die offiziell vom Hersteller freigegebene Firmware verwendet. Diese könnte womöglich vom Funktionsumfang bzw. den Hintergrundprozessen etwas schlanker ausfallen als die Android Version des Nexus 5, wodurch die Unterschiede in der Ausführungszeit entstehen können. Beobachten kann man aber auch, dass

[1] Der besseren Übersicht halber sind die Namen der Geräte, sowie die der verwendeten FastCV Modi abgekürzt dargestellt. L. P. = Low Power, P. = Performance, CPU O. = CPU OFFLOAD

bei der Videosequenz die Verwendung von FastCV immer zu einer besseren Performance, also zu einer kürzeren Ausführungszeit, geführt hat. Im Maximum bis zu 1.57x so schnell und im Mittel 1.34x so schnell. In Abbildung 4.2 ist dieser Zusammenhang grafisch dargestellt.

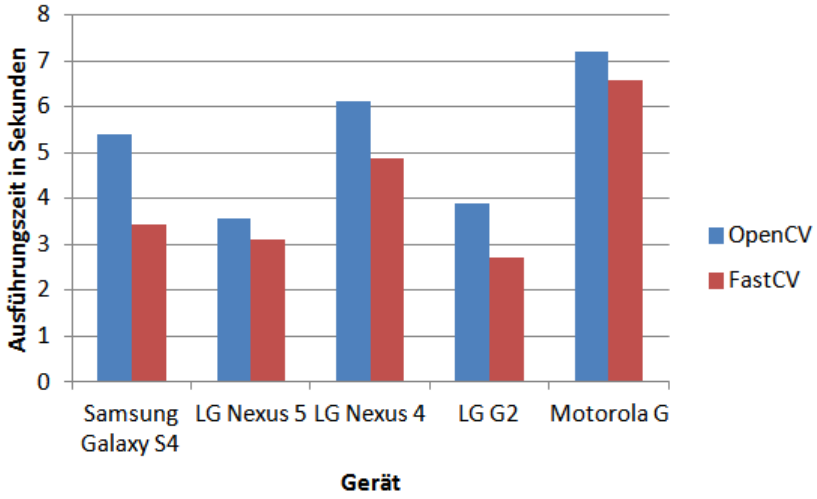


Abbildung 4.1: Vergleich der einzelnen Ausführungszeiten zwischen FastCV und OpenCV auf Geräten mit Qualcomm Chip. Man sieht deutlich, dass die Nutzung von Qualcomm Chips auf jedem Gerät Vorteile mit sich bringt. Je nach Gerät fällt der Vorteil deutlicher (z. B. im Falle eines Galaxy S4) oder weniger deutlich (im Falle eines Motorola G oder LG Nexus 5) aus.

Nach der Diskussion und dem Vergleich der Ergebnisse der einzelnen Geräte, soll nun natürlich auch die Leistungsfähigkeit gegenüber der alten Plattform, dem Gumstix Overo Fire sowie einem handelsüblichen X86 PC betrachtet werden. Für den Vergleich wird das LG G2, welches die kürzeste Ausführungszeit bei der Testsequenz hat, herangezogen. Nominell, von der reinen CPU Geschwindigkeit ausgehend gesehen, bietet das LG G2 mit 2.26 Ghz getaktetem Prozessor 3.14x soviel Rechenleistung wie der Overo Fire und nur knapp weniger Rechenleistung als der getestete X86 Prozessor. Wie man anhand der Tabelle 4.3 sehen kann, darf aber nicht nur die reine Prozessorgeschwindigkeit in Betracht gezogen werden, sondern es muss auch die Leistung einer neuen Architektur mit einbezogen werden. Dadurch fällt der Unterschied noch größer aus - das Smartphone ist dann nicht 3.14x so schnell wie der Gumstix, sondern ganze 7.3x schneller bei der Videosequenz. Mit einem Geschwindigkeitsvorteil des Smartphones gegenüber dem Gumstix musste bereits im Vorfeld anhand der Hardwareunterschiede gerechnet werden. Etwas überraschend war ist jedoch der Geschwindigkeitsunterschied zwischen dem LG G2 und dem X86 PC. Hierbei ist das Smartphone immerhin noch 1.8x so schnell, obwohl der X86 PC nominell gesehen eine schnellere Hardware haben müsste. Weiters ist zu beobachten, dass die Ausführungszeit des Optischen Fluss Algorithmus auch am X86 durch den Einsatz von FastCV Geschwindigkeitsvorteile mit sich bringt. Diese Ergebnisse sprechen deutlich für die Leistungsfähigkeit des Smartphones als Einsatz für die Drohne.

Betrachtet man die Ergebnisse, kann man erkennen, dass sich der Einsatz von FastCV in Kombination mit OpenCV durchaus lohnt. Durch diesen Umstand ist es also möglich, Funktionen, die einen deutlichen Performancevorteil mit sich bringen, für FastCV zu verwenden und z.B. die restliche App durch den großen Umfang an Funktion von OpenCV schneller zu entwickeln. Abschließend kann man sagen, dass sich der Einsatz eines Smartphones als Drohnennplattform in

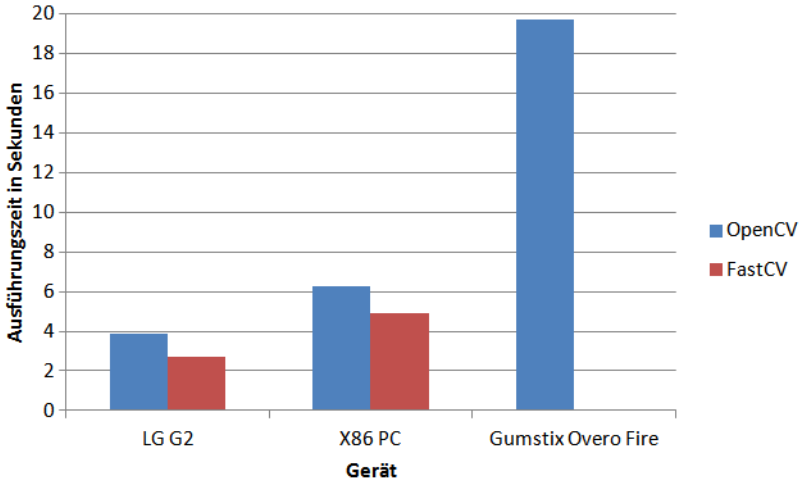


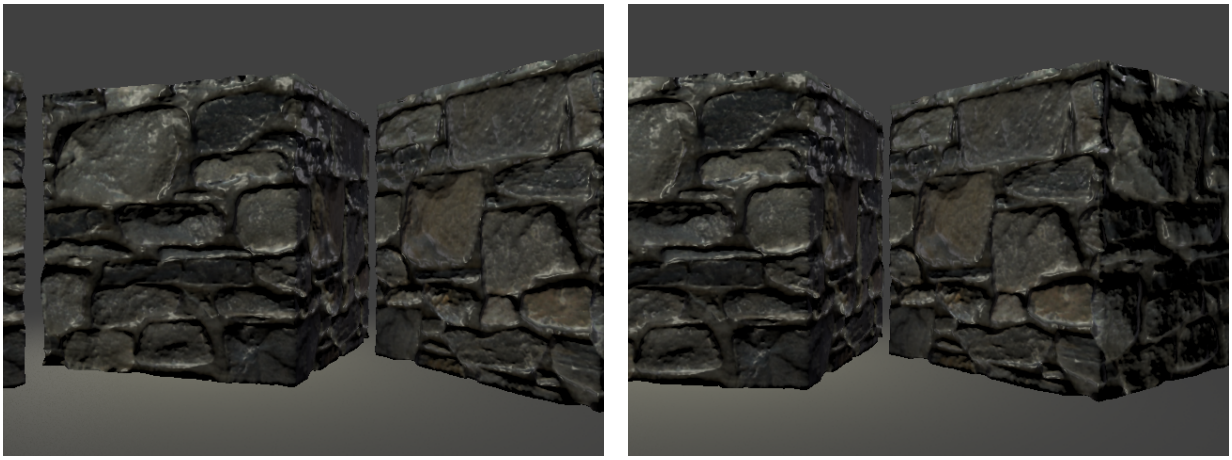
Abbildung 4.2: Vergleich der Ausführungszeiten zwischen einem Smartphone, der alten Drohnenplattform - dem Gumstix, sowie einem X86 PC. Man sieht, dass das Smartphone deutlich schneller als die Gumstix Overo Fire und überraschend auch etwas schneller als der X86 PC ist. Da die Ausführungszeit von FastCV am Gumstix nicht getestet wurde, fehlt der entsprechende Balken im Diagramm.

Verbindung mit einem Mono-Kamera-System (welches das Smartphone bereits integriert hat) gut eignet und, dass die Anforderungen, die an die Hardware gestellt werden, dadurch bestmöglich abgedeckt werden können. Gerade im Heimanwender-Bereich hat das Smartphone als Drohnenplattform eine noch signifikantere Rolle, da man relativ simpel sein aktuelles Smartphone mit der Drohne verbinden und so eine App starten kann. Eine andere Möglichkeit bzw. ein anderer Anwendungsfall wäre aufgrund der schnelllebigen Hardware folgender: ein ein bis zwei Jahre altes Gerät fällt auf den Boden und das Display springt und die Reparatur dessen lohnt sich nicht mehr - für die Drohne kann man dies aber noch verwenden. Man hat somit ein kostengünstiges Gerät, dass diverse Sensoren, eine gute Kamera und auch einiges an Performance mit sich bringt, zur Verfügung.

4.2 Rotationskompensation

Die Rotationskompensation soll die Drehung um die Optischen Achsen ausgleichen und soweit wie möglich entfernen bzw. neutralisieren um nur noch eine Translationsbewegung zu erhalten. Dazu wurde der 2. Ansatz, die Rotation über die Pose Estimation zu bestimmen, ausgewählt und verwendet. Im Vergleich zu einem Hardwaresensor, wie etwa einem Gyroskop, wird eine Messung nur zwischen jeweils 2 Bildern durchgeführt. Dies bedeutet, dass die Frequenz der Messung bei weitem nicht so hoch ist wie bei einem Gyroskop. Während moderne Gyroskop eine Frequenz von 200Mhz oder teilweise noch mehr aufweisen, hängt die Frequenz der Pose Estimation von mehreren Faktoren ab. Die Bildgröße bzw. die zu berechnenden Punkte haben hier den größten Einfluss, da hoch aufgelöste Bilder einen höheren Berechnungsaufwand haben und der Algorithmus dadurch pro Bild länger benötigt. Weiters kommt hinzu, wie schnell die restlichen Algorithmen ausgeführt werden können, also im Prinzip wie viele Frames per Second (FPS) erreicht werden. Umso höher die FPS sind, umso höher wird automatisch die Frequenz, wobei die Hardwareeinschränkungen der Kamera (maximal mögliche FPS) bestehen bleiben. Diese muss auch in

der Lage sein, mehr als z.B. 30 FPS zu liefern. Für die Pose Estimation bzw. allgemein für den Ablauf der Hinderniserkennung muss man einen Mittelweg gehen, da einerseits der Algorithmus schnell genug, aber andererseits die Rotationskompensation auch genau genug sein soll. Man muss daher bis zu einem sinnvollen Grad Ausreißer finden und entfernen, ohne die Ausführungszeit zu weit in die Höhe zu treiben. Anders als bei der Rotationserkennung mit einem Gyroskop, ist es bei der Pose Estimation rein über die Mono-Kamera nicht mehr nötig das Kamerabild mit dem Gyroskop zu synchronisieren. Die Pose Estimation wurde durch synthetische Szenen (Siehe Abbildung 4.3), sowie durch die Verwendung eines handelsüblichen Modellbauservos, auf dem die Kamera montiert wurde, getestet, um auch Bilder einer realen Kamera zur Verfügung zu haben und diese mit der synthetischen Szene vergleichen zu können. Für die synthetische Szene wur-



(a) Bild 1 (b) Bild 30

Abbildung 4.3: Beispielhaft 2 Bilder der synthetischen Testszene von Würfeln, welche mit einer Steintextur versehen sind um ein besseres Erkennen von Features zu ermöglichen. Die Kamera bewegt sich mit 0.333 ° zwischen den jeweils zwei Bildern. Zwischen Bild 1 und Bild 30 sind 10° Unterschied. Die Kamera führt dabei eine Bewegung von links nach rechts aus.

den drei Würfel mit einer Steintextur (da Flächen ohne Textur nicht für das Tracking geeignet sind) verwendet und eine Kamerabewegung um die Optische x-Achse mit einer Geschwindigkeit von 10 Grad pro Sekunde durchgeführt. Diese Bewegung entspricht einer Yaw Bewegung, wobei zwischen jeweils zwei Bildern genau 0.333 ° Differenz besteht. Die Ergebnisse der Messung sind in Abbildung 4.4 zu sehen. Um die realen Bilder einer Drehung aufzunehmen, wurde auf einem Modellbauservo eine Platte fixiert, auf welcher die Kamera einen stabilen und sicheren Halt hat. Der Servo ist direkt mit einem Arduino Duemilanove [5] verbunden. Auf dem Arduino wird die Servo Library [47] verwendet, um den Servo anzusteuern und ihn mit einer Geschwindigkeit von 0.66 Grad pro Sekunde zu drehen. Da die Pose Estimation annähernd mit den realen Werten der Rotation des Servos übereinstimmt, ist es im nächsten Schritt notwendig, die Rotation aus den Optischen Fluss Vektoren zu entfernen. Stellvertretend für die Effektivität der Rotationskompensation um alle Achsen, wurde die Block Szene in Abbildung 4.3 verwendet, bei der die Kamera eine Rollbewegung durchführt. Außerdem wurde eine reale Szene verwendet in der eine Yaw Bewegung (Drehung um die optische y-Achse) durchführt wurde. In Abbildung 4.5, sowie in Abbildung 4.6, werden die Bewegungen bzw. die Optischen Fluss Vektoren vor sowie nach der Rotationskompensation gegenübergestellt, um die Effektivität zu verdeutlichen.

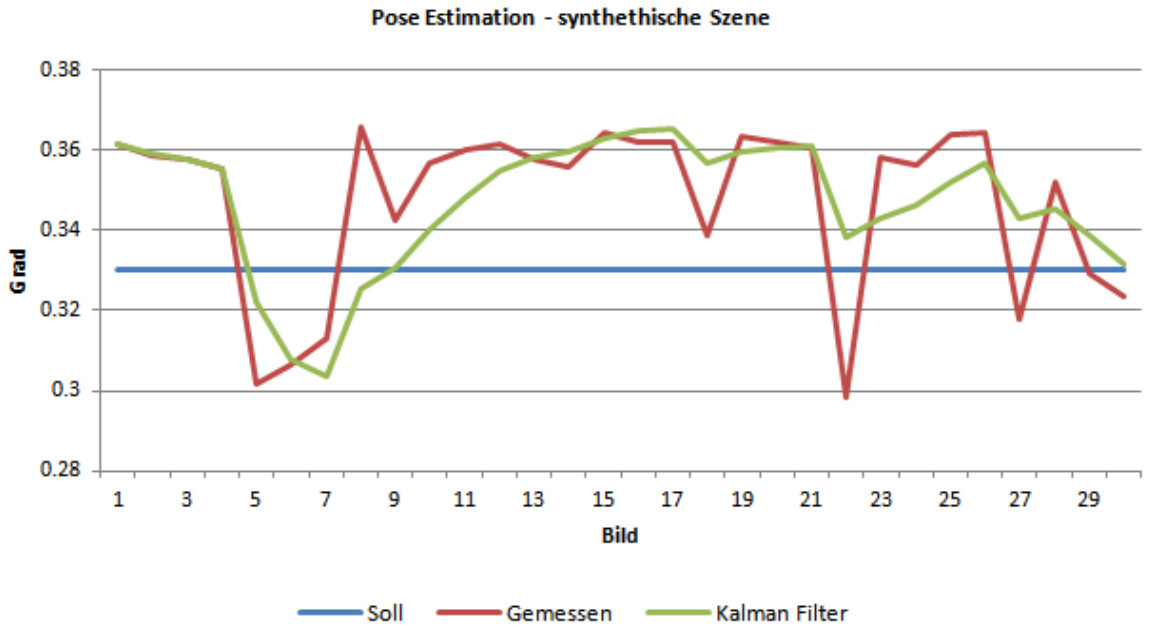
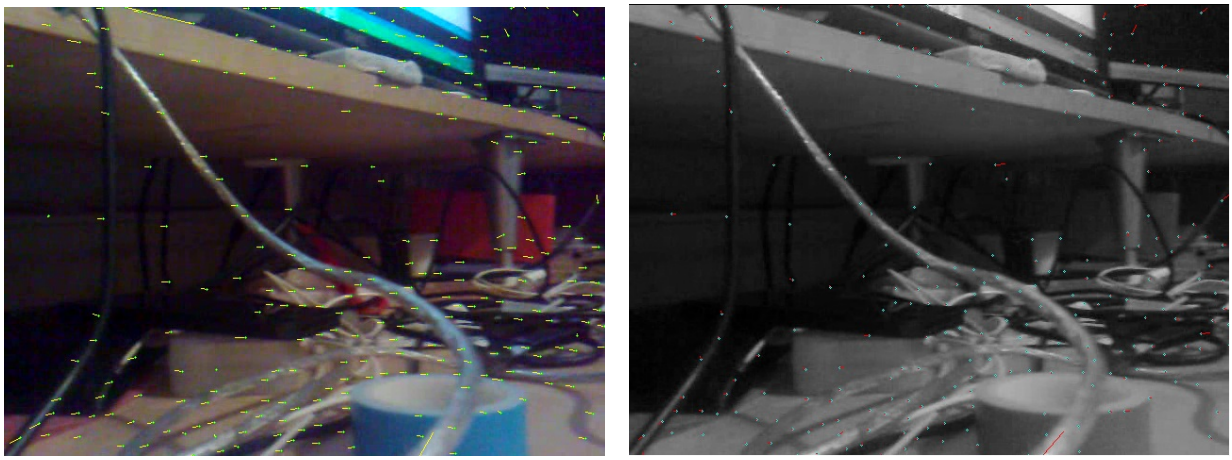


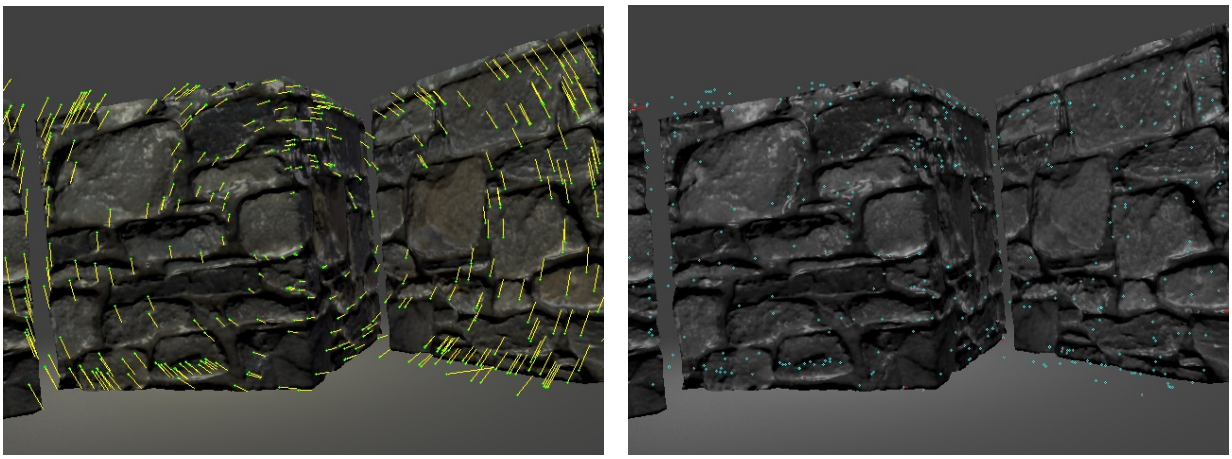
Abbildung 4.4: Messergebnisse der Pose Estimation. In der Grafik sind die realen Werte, die gemessenen Werte, sowie die mit einem Kalmanfilter gefilterten Werte der Kameradrehung eingezeichnet. Anders als es auf den ersten Blick, durch die Skalierung der Achsen erscheint, beträgt die Abweichung vom Soll Wert (Rotation) im Maximum nur 0.029°.



(a) Optische Fluss Vektoren vor der Rotationskompensation

(b) Optische Fluss Vektoren nach der Rotationskompensation

Abbildung 4.6: Eine Yaw Bewegung der Kamera. Dabei wurde eine Drehung um 0.66 Grad zwischen jeweils 2 Bildern durchgeführt. In Abbildung 4.6 (a) sind die ungefilterten Vektoren des Optischen Flusses gelb eingezeichnet. In Abbildung 4.6 (b) sind die gefilterten Vektoren des Optischen Flusses zu sehen. Hierbei lässt sich erkennen, dass die Rotation zum Großteil entfernt werden konnte und somit der Fehler bei einer absoluten Entfernungsmessung minimiert wird. Das Ergebnis ist wie zu erwarten, aufgrund der etwas weniger exakten Pose Estimation, etwas schlechter als bei der synthetischen Szene.



(a) Optische Fluss Vektoren vor der Rotationskompensation

(b) Optische Fluss Vektoren nach der Rotationskompensation

Abbildung 4.5: Eine Rollbewegung der Kamera innerhalb der synthetischen Block Szene, dabei wurde eine Drehung um 5 Grad zwischen den beiden Frames durchgeführt. In Abbildung 4.5 (a) sind die ungefilterten Vektoren des Optischen Flusses gelb eingezeichnet. Während man in Abbildung 4.5 (b) die gefilterten Vektoren des Optischen Flusses sieht. Es ist deutlich zu erkennen, dass die Rotation sehr gut entfernt werden konnte.

4.3 Focus of Expansion

Der Focus of Expansion gibt, wie in Kapitel 3 bereits beschrieben, die Richtung an in die sich die Drohne bewegt. Dabei ist wichtig, dass dieser genau bestimmt wird, da der FoE in weiterer Folge ein Grundbestandteil der Hinderniserkennung ist. Wird der FoE nicht korrekt bestimmt bzw. weicht die Berechnung zu sehr von dem wirklichen Wert ab, ist auch die Berechnung der Entfernung zu einem Hindernis fehlerhaft. Der FoE wird nur berechnet, wenn eine Vorwärts- oder Rückwärtsbewegung ausgeführt wird. Die Tests haben gezeigt, dass die Berechnung des FoE instabil ist. Aus diesem Grund wurden zwei Maßnahmen durchgeführt, um den FoE zumindest etwas stabilisieren zu können. Einerseits wurde ein Filter verwendet, der den Mittelwert der letzten 5 Positionen verwendet, andererseits wurden in einem zweiten Schritt die längeren Vektoren des Optischen Flusses stärker gewichtet. Die Gewichtung der Vektoren wurde deswegen durchgeführt, da lange Vektoren einen größeren Einfluss auf die Position des FoE haben als kurze, und so der FoE genauer bestimmt werden kann.

Um die Effizienz der FoE Bestimmung sowie jene der zwei zusätzlichen Maßnahmen zu eruieren, wurden sowohl Tests mit einer synthetischen Videosequenz, als auch mit einer realen Videosequenz durchgeführt. Bei der synthetischen Videosequenz bewegt sich die Kamera direkt in Richtung der zwei Blöcke. Da die Videosequenz am Computer erstellt wurde, hat diese keinerlei Rotation und auch keinerlei Wackeln wie es bei realen Videosequenzen bzw. bei einer Drohne unvermeidbar ist. Dadurch ist auch zu erwarten, dass die Ergebnisse bei der synthetischen Videosequenz deutlich besser ausfallen werden als die bei den realen Sequenzen.

In Abbildung 4.9 sieht man den berechneten FoE einmal mit Filter und Gewichtung, sowie einmal ohne Filter und Gewichtung. Es ist zu erkennen, dass die Maßnahmen zu einer Verbesserung der FoE Bestimmung geführt haben. Aber selbst mit Filter und Gewichtung, ist eine Abweichung vom Soll vorhanden. Im Durchschnitt beträgt die Abweichung des gefilterten FoE Wertes auf

der x-Achse 4.11 % und auf der y-Achse 0.71 %. Jedoch liegt die Abweichung im Maximum auf der x-Achse bei 19 % und auf der y-Achse bei 9.7 %. Der Filter hilft zwar, dass der FoE etwas weniger von der Soll Position abweicht, jedoch sind die absoluten Abweichungen zu groß und beeinträchtigen somit in weiterer Folge auch die Time to Contact (TTC) Berechnung und die Bestimmung der absoluten Entfernung. Die Ursache dafür sind kleinere Abweichungen in der Rotationskompensation. Während der FoE Testsequenz wurden im Mittel ein Fehler von 0.036 ° für die x-Achse, 0.041 ° für die y-Achse und -0.009 ° für die Z Achse ermittelt. Diese geringe Abweichung reicht bereits aus, um die FoE Berechnung zu verfälschen.

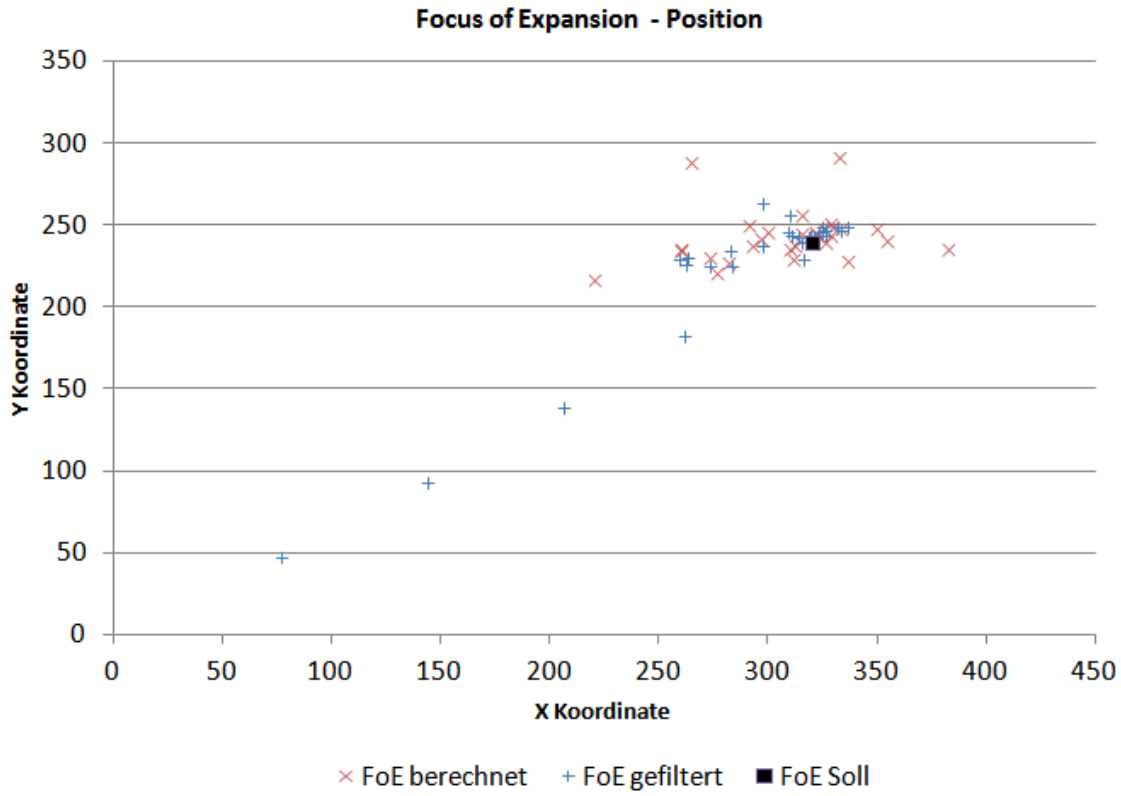


Abbildung 4.7: Messergebnisse der FoE Berechnung. In der Grafik ist zu Beginn zu erkennen, dass einige Punkte des FoE Filters weit entfernt vom Soll liegen. Dies hat damit zu tun, dass der Filter in dieser Zeit initialisiert wird. Man sieht weiters, dass durch die Gewichtung, sowie durch den Filter der FoE näher am Soll liegt, also ohne diese Maßnahmen.

Zusätzlich zur synthetischen Szene wurde der FoE der Gartenszene (Abbildung 4.8) ebenfalls ermittelt. Bei der Gartenszene wurde der Test durchgeführt, indem die Kamera in der Hand so ruhig wie möglich gehalten wurde und sich dann vorwärts in Richtung der im Bild ersichtlichen Sandkiste bewegt wurde. Diese Bewegung ist mit der einer Drohne vergleichbar. Sie beinhaltet leichte Rotationen, sowie Translationen, also kurz gesagt ein Wackeln der Kamera. Je nachdem wie gut der PID Regler für die Stabilisierung der Drohne umgesetzt wurde, kann mehr oder weniger Wackeln entstehen. Die Ergebnisse der realen Szene zeigen deutlich, dass der FoE hier noch stärker vom Soll abweicht, als zuvor bei den Messungen der synthetischen Szene. Eine gewisse Abweichung vom Soll wäre vertretbar, solange diese sich stabil in einem Bereich befindet. Verändert jedoch der FoE selbst mit dem angewandten Filter zwischen zwei Bildern seine Position

zu stark, dann ist in weiterer Folge die Bestimmung der absoluten Distanz mit einem zu großen Fehler behaftet.

(a) Bild 1 (b) Bild 35 (c) Bild 70

Abbildung 4.8: Drei Bilder einer realen Gartenszene, in der sich eine Drohne bewegen könnte. Dabei geht die Bewegung direkt auf die Sandkiste zu. In der Szene ist auch die Wand in der rechten Bildhälfte zu beachten. Diese Wand ist für das Featuretracking besonders schwierig da es bis auf den Abschnitt wo die weiße auf die graue Farbe kommt keine markanten Punkte zu sehen sind welche getrackt werden könnten.

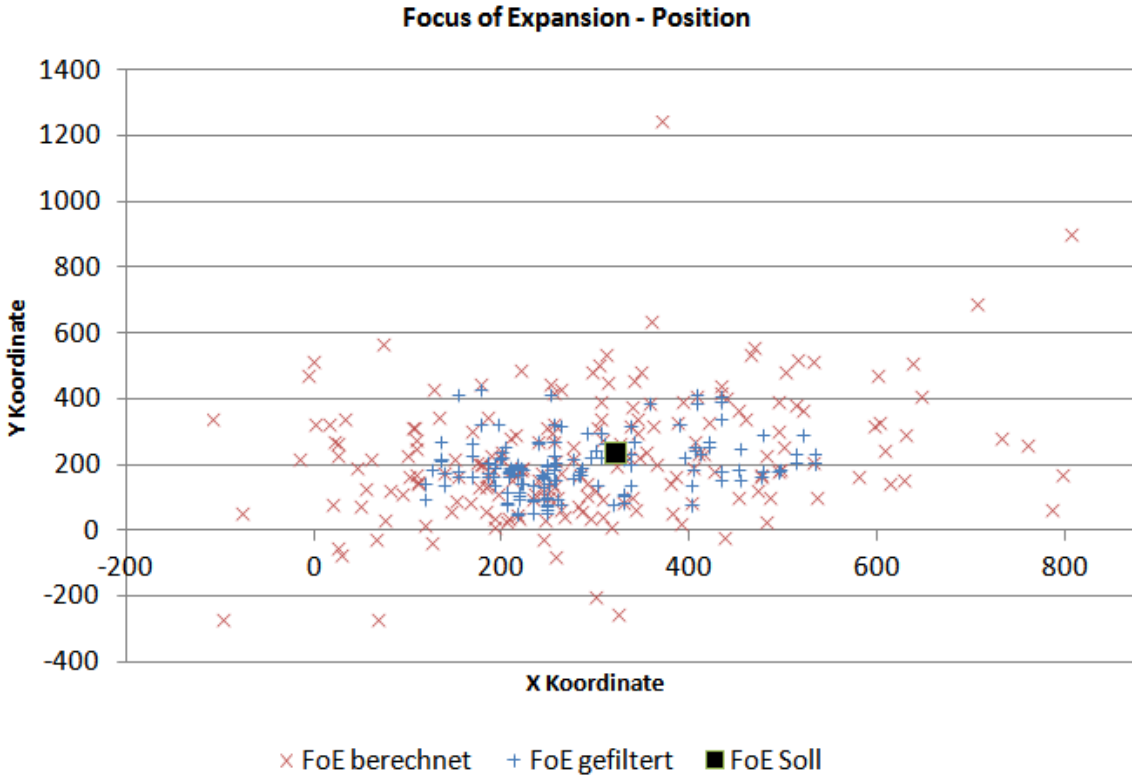


Abbildung 4.9: Messergebnisse der FoE Berechnung der Gartenszene. Wie man anhand der Grafik erkennen kann, trägt der Filter zu einer Verbesserung der Ergebnisse bei. Dennoch streut die FoE Position ziemlich stark um den Soll Wert. Dies führt in weiterer Folge zu einem großen Fehler bei der Berechnung der absoluten Distanz.

4.4 Time to Contact

Für die Berechnung der Time to Contact (TTC) ist der Focus of Expansion, sowie die Länge und die Endpunkte der Optischen Fluss Vektoren relevant. Dabei wird der Weg von den Endpunkten der Vektoren bis zum FoE berechnet. Die Bestimmung des TTC ist im eigentlichen Sinne nur ein Verhältnis, das den Weg bestimmt und kann nicht optimiert werden, da die TTC Werte rein vom FoE sowie von den Werten der Vektoren des Optischen Flusses abhängen. Sind diese Werte fehlerbehaftet, sind die Werte des TTC ebenso fehlerbehaftet.

Für die TTC Berechnung wurde ein fixer Punkt mit den Koordinaten X: 390 Y: 170 zu den automatisch getrackten Features der synthetischen Sequenz hinzugefügt. Dieser Punkt wurde hinzugefügt um immer die gleiche Referenz zu haben, da die Features ansonsten "wandernünd somit das Berechnung beeinflussen. Das Ergebnis ist in Abbildung 4.10 zu sehen. Dabei ist zu beachten, dass der Graph am Anfang steil verläuft, da in dieser Zeit der FoE Filter initialisiert wird. In realen Bedingungen ist dies zu vernachlässigen, da in der Zeit, in der die Drohne für den Start vorbereitet wird bzw. kurz vor Start ist, genug Zeit bleibt um den Filter entsprechend zu befüllen.

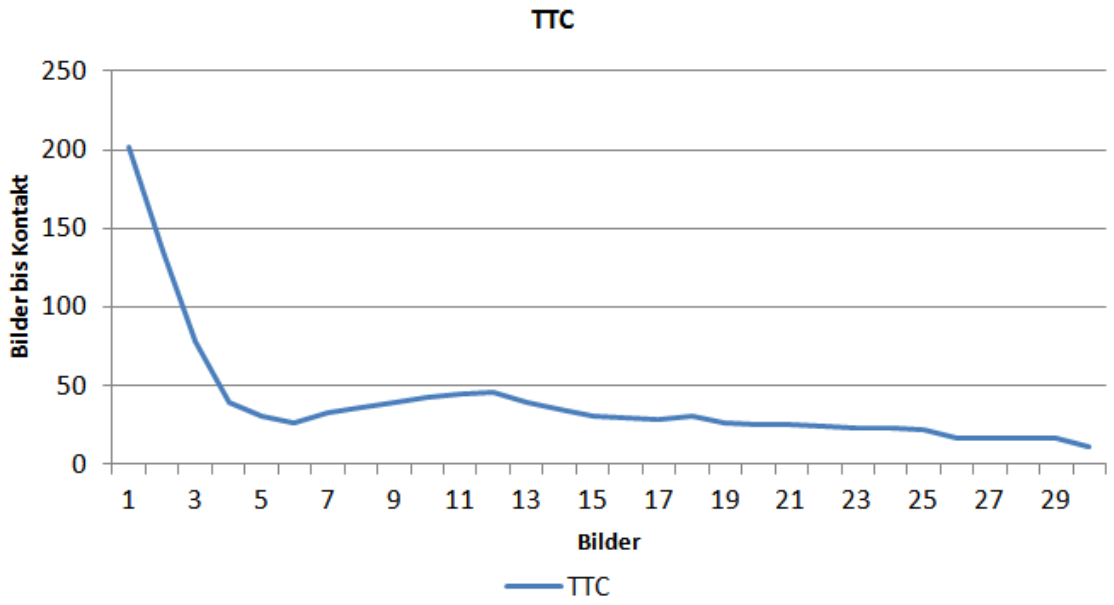


Abbildung 4.10: Messergebnisse der TTC Berechnung. In der Grafik ist zu Beginn zu erkennen, dass der Graph stark abfällt. Dies passiert während der FoE Filter Initialisierung. Die Linie flacht danach ab und man kann den Trend erkennen. Je näher man dem Hindernis, also in diesem Fall den Würfeln, kommt, desto niedrigerer wird die TTC. Um die Bilder 6 bis 13 steigt der TTC jedoch obwohl dieser fallen müsste. Dies kann z.B. auf einen nicht 100% stabilen FoE zurückgeführt werden.

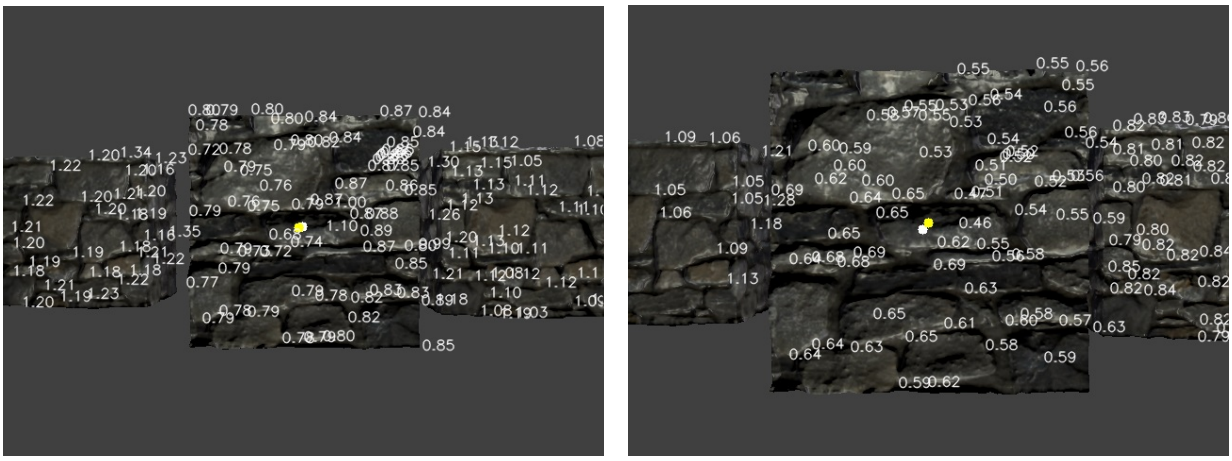
Die TTC bei kurzen Vektoren ist im Bereich um den FoE sehr anfällig für Fehler. Angenommen der FoE befindet sich im Punkt X: 320, Y: 240 und der zu verfolgende Vektor hat den Endpunkt auf Punkt X: 330, Y: 240 und eine Länge von 1. Wird die Rotation nun falsch berechnet, kann es vorkommen, dass der Vektor nun angenommen die Länge 5 hat. Dies würde das Ergebnis sehr verfälschen. Dasselbe gilt für den FoE; ist dieser nicht stabil genug, wird die Berechnung ebenfalls verfälscht. Da man bei der TTC selbst jedoch, wie bereits angesprochen, nichts verbessern kann,

ist man nur von der Rotationskompensation, sowie der FoE Bestimmung abhängig.

4.5 Hinderniserkennung

Die Hinderniserkennung im Allgemeinen, also die Bestimmung der absoluten Entfernung zu einem Hindernis, hängt von der Qualität der bereits besprochenen Punkte ab. Die TTC bestimmt dabei nur die relative Entfernung zu einem Hindernis. Will man die absolute Entfernung bestimmen, ist zudem die aktuelle Geschwindigkeit der Drohne nötig. Diese kann z.B. entweder über Visuelle Odometry oder GPS bestimmt werden. Bei den durchgeführten Tests wurde jedoch die Geschwindigkeit manuell bestimmt und dann als Vorgabe im Programm verwendet.

Dabei wird wieder eine Synthetische Szene mit den Würfeln bzw. Blöcken verwendet. Eine reale Szene wird hier bewusst nicht verwendet, da die FoE Berechnung bereits zu ungenau ist und somit auch die Bestimmung der absoluten Distanzen zu einem Hindernis nicht verlässlich erfolgen kann. In Abbildung 4.11 sind 2 Bilder der Synthetischen Testsequenz mit den überlagerten Entfernungswerten zu sehen. Dabei ist deutlich zu erkennen, dass die Entfernung zum mittleren Block am geringsten ist, während der rechte bereits etwas weiter entfernt steht und der im Bild linke Block noch weiter entfernt ist als die anderen zwei Blöcke. Dies entspricht auch der realen Anordnung der Blöcke in der Testsequenz. Weiters erkennt man auch, dass die Werte pro Block relativ einheitlich sind, d.h. so gut wie keine Ausreißer vorhanden sind.



(a) Bild 15 der Synthethischen Szene

(b) Bild 23 der Synthethischen Szene

Abbildung 4.11: Beispielhaft 2 Bilder der synthetischen Testszene von Würfeln auf die sich die Kamera frontal zubewegt. Dabei werden die gemessenen Entfernungswerte als Überlagerung im Bild blau dargestellt. Man sieht deutlich, dass die gemessenen Entfernungen der Blöcke mit der Anordnung dieser übereinstimmt - der mittlere ist am nächsten gelegen, während der rechte Block etwas dahinter ist und der linke noch weiter entfernt. Weiters ist im Bild der FoE, einmal als gelber und einmal als weißer Punkt zu sehen, dies entspricht den gefilterten sowie berechneten FoE.

Die Bestimmung der absoluten Entfernung zu Hindernissen ist leider aufgrund der vorhergehenden Berechnungen des FoE und der Rotation nicht sehr stabil, wodurch es immer wieder zu falschen Ergebnissen bei der Bestimmung der absoluten Entfernung kommt.

5 Ausblick

In dieser Arbeit wurde versucht, eine Hinderniserkennung für die ICT Drohne mittels eines dafür geeigneten Systems zu bewerkstelligen. Als dafür gut geeignete Systeme haben sich Mono- und Stereo-Kamera-Systeme herausgestellt, da diese nach eingehender Recherche Vorzüge gegenüber anderen Systemen hatten. Dies wurde ausführlich in Kapitel 2 diskutiert. Für die Umsetzung der Hinderniserkennung wurde ein Mono-Kamera-System verwendet, mit dem Optischen Fluss als Hauptbestandteil.

Während der Umsetzung der Aufgabe stellte sich heraus, dass das alte Computersystem der ICT Drohne nicht den Ansprüchen an die Laufzeiten der Algorithmen gerecht werden konnte, vor allem da kein Zugriff auf die TI Vision Library gewährleistet wurde. Aus diesem Grund musste ein neues System gefunden werden, welches das alte Computersystem ersetzen kann. Als gutes System präsentierte sich ein Mobiltelefon als Plattform, da dieses alle gestellten Anforderungen am besten erfüllen kann. Dass Mobiltelefone für die Hinderniserkennung oder auch für die Navigation in Räumen gut eingesetzt werden können unterstreicht die derzeitige Forschung von Google mit dem Projekt Tango [46]. Bei dem Projekt Tango werden ebenfalls Mobiltelefone sowie Tablets verwendet, um eine 3D Darstellung der Umgebung zu erzeugen. Gerade auch im Bereich von nicht kommerziellen Projekten,überzeugen Mobiltelefone als Hardwareplattform für autonom gesteuerte Geräte, egal ob Boden- oder Luftfahrzeuge. Ihr Vorteil ist die rasant wachsende Leistung und der, durch die Massenproduktion, günstige Preis. Dies wird vor allem bei Umständen wie beispielsweise einer Vertragsverlängerung mit neuem Mobiltelefon interessant, da man dann das alte Mobiltelefon als sehr kostengünstige und leistungsfähige Plattform zur Hinderniserkennung einsetzen kann. Die Leistung moderner Smartphones wurde ebenfalls in Kapitel 3 untersucht und die Ergebnisse in Kapitel 4 dargestellt.

Während der Umsetzung der Hinderniserkennung stellte die Rotationskompensation eine große Herausforderung dar, diese konnten zwar zum größten Teil, aber nicht in ihrem vollen Ausmaß, gelöst werden. Die Rotation verändert die Längen der Vektoren des Optischen Flusses je nachdem wie stark die Rotation ist und verfälscht so die Entfernungsmessung. Dieser Fehler konnte durch die Bestimmung der Rotation über die Pose Estimation soweit minimiert werden, dass dieser Einfluss auf die Berechnung vernachlässigt werden kann. Jedoch ist, wie man anhand der Ergebnisse in Kapitel 4 sehen kann, die Focus of Expansion- Bestimmung extrem anfällig für Abweichungen in der Rotation. Während bei reiner Translation noch eine relativ gute Bestimmung des FoE möglich war, ist dies bei realen Szenen nicht mehr der Fall. Dies hat mehrere Gründe, die hier kurz aufgezählt und anschließend diskutiert werden. Ebenfalls werden Verbesserungsmöglichkeiten bzw. Erweiterungen aufgezählt:

- Die ICT Drohne bzw. Multicopter allgemein haben 6 Freiheitsgrade
- Schlechte Qualität der Kamera (z. B. Rauschen)
- Geringe Geschwindigkeit bei autonomen Drohnen
- Tradeoff zwischen Genauigkeit und Geschwindigkeit der Berechnungen
- Bestimmung der Rotation überaus wichtig
- Bewegung der Drohne ist notwendig
- Keine Bewegung innerhalb der Szene
- Bestimmung der Geschwindigkeit u. Höhe

Die ICT Drohne bzw. Multicopter allgemein haben 6 Freiheitsgrade

Dadurch, dass die ICT Drohne ein Multicopter ist, hat diese 6 Freiheitsgrade. Dies bedeutet, dass sie sich nach Oben/Unten, Links/Rechts, Vor- und Zurück bewegen, sowie auch um ihre Achsen drehen kann. Ein Multicopter ist sozusagen der am schwierigsten umzusetzende Fall einer Hinderniserkennung mittels Optischen Fluss.

Hat man ein Flächenmodell anstatt eines Multicopters, kann sich dieses Modell zwar ebenfalls um alle Achsen drehen, jedoch entfallen die Bewegungen nach Hinten, sowie Links/Rechts und auch die Bewegung nach oben und Unten kann zwar durchgeführt werden, ist aber immer mit einer Vorwärtsbewegung kombiniert. Dadurch, dass ein Flächenmodell eine Mindestgeschwindigkeit braucht um überhaupt in die Luft zu kommen und auch um dort zu bleiben, ist immer ein großer Anteil der Bewegung eine Vorwärtsbewegung. Dies hilft den FoE zu bestimmen, da die Vektoren des Optischen Flusses dadurch automatisch länger und so auch stabiler sind als wenn sich eine Drohne langsam vorwärts bewegt.

Ähnlich verhält es sich bei Bodenfahrzeugen, nur dass diese noch mehr Einschränkungen mit sich bringen, womit die Bestimmung des FoE noch etwas weniger fehleranfällig wird. Ein Bodenfahrzeug kann sich nicht nach Oben/Unten u. Links/Rechts bewegen, außerdem sind die Rotationen um die Achsen deutlich eingeschränkt, da sich ein Bodenfahrzeug nur um die Optische z-Achse drehen kann. Die Drehung wird ebenfalls wieder mit einer Mindestgeschwindigkeit verbunden, wenn auch diese geringer ausfallen kann als bei Flächenmodellen.

Ein weiteres Problem bei der FoE Bestimmung ist, dass dieser nur für Vorwärts bzw. Rückwärtsbewegungen (dann jedoch Focus of Contraction genannt) ermittelt werden kann. Bewegt sich die ICT Drohne also nur Links/Rechts oder Rauf/Runter, ist es nicht möglich den FoE zu bestimmen, da dieser bei solchen Bewegungen nicht existiert.

Daraus kann man schließen, dass die Hinderniserkennung für Bodenfahrzeuge und Flächenmodelle besser geeignet wäre bzw. allgemein durch die Einschränkung in der Bewegung deutlich weniger fehlerbehaftet sein kann als dies bei Multicoptern der Fall ist.

Schlechte Qualität der Kamera (z. B. Rauschen)

Bei der Entwicklung der Hinderniserkennung stellt sich heraus, dass die Berechnung auch stark von der Kamera abhängig ist. Eine handelsübliche Webcam durchaus zum Teil für z. B. Videotelefonie geeignet, aber nicht für den Einsatz als System für eine Drohne. Dies hat damit zu tun, dass man zwar durch Kalibration die Verzerrungen recht gut entfernen kann, jedoch Rauschen oder aber auch ein schlechter Fokus auf die verschiedenen Objekte einer Umgebung, das Tracking der Features und somit auch die Berechnung der Entfernung fehlerhaft machen. Beim Einsatz eines Mobiltelefons als Plattform stellt dies jedoch ein eher kleines Problem dar, da moderne Mobiltelefone meist eine gute Kameraqualität bieten und zudem auch teilweise bereits einen Bildstabilisator eingebaut haben.

Geringe Geschwindigkeit bei Autonomen Drohnen

Bereits im 1. Punkt wurde dieser Aspekt angesprochen. Ist die Geschwindigkeit der Drohne zu klein, bzw. überwiegt der Anteil an Vorwärtsbewegung nicht, kann dies dazu führen, dass die Bestimmung des Focus of Expansion sehr ungenau wird. Dies hat damit zu tun, dass, wenn die Längen der Optischen Fluss Vektoren bereits im Subpixel Bereich sind, sehr anfällig auf Fehler bei der Rotationsbestimmung sind. Eine Mindestgeschwindigkeit von etwa 0.5 m/s sollte daher eingehalten werden.

Tradeoff zwischen Genauigkeit und Geschwindigkeit der Berechnungen

Für die Bestimmung der Entfernung ist es wichtig, dass es so gut wie keine Ausreißer gibt, die das Ergebnis verfälschen. Dies ist für alle Stufen des Programms wichtig, begonnen bei der Pose Estimation bis hin zur Bestimmung der Entfernung. Da die Ausführungszeit des Algorithmus von entscheidender Bedeutung ist und man versucht, die Frames per Second so nahe wie möglich an die maximal von der Kamera zur Verfügung gestellten FPS zu bringen, muss man einen Mittelweg zwischen Genauigkeit und Geschwindigkeit finden. Man kann zwar viele Filterstufen einbauen, einen Bruteforcematcher oder ähnliches verwenden, nur kosten diese viel Zeit und verlangsamen dadurch die Ausführungszeit des Algorithmus. Während vielleicht auf High End X86 Hardware oder auf GPGPU Systemen solche Verfahren verwendet werden können, würden diese auf einem Mobiltelefon, welches als Plattform für die Drohne gedacht ist, zu viel Rechenzeit in Anspruch nehmen. Auch wenn die mobile Plattform schneller ist, als das zur Verfügung gestandene X86 Notebook, ist dieses von der Rechenleistung natürlich deutlich langsamer als ein gerade erst auf den Markt gekommenes neues System mit High End CPUs und GPUs.

Bestimmung der Rotation

Dass die Bestimmung der Rotation einen extrem großen Einfluss auf die Berechnung des FoE hat, war in diesem Ausmaß nicht zu erwarten. Während die Längen der Optischen Fluss Vektoren sehr gut angenähert werden konnten, verfälscht eine Abweichung von weniger als 0.1 Grad/s die Berechnung des FoE deutlich. Dies kann man auch sehr gut anhand der Ergebnisse in Kapitel 4 nachvollziehen. Als Verbesserung des Systems sollte deswegen auch ein Gyroskop verwendet und versucht werden, die Bilder mit den Messungen zu synchronisieren. Auch eine Sensorfusion der Gyroskop Daten mit jenen der Pose Estimation wäre eine Überlegung wert, sofern sich bei Tests zeigt, dass dies eine noch bessere Bestimmung der Rotation als nur über die Pose Estimation oder nur über ein Gyroskop mit sich bringt.

Bewegung der Drohne ist notwendig

Wie bereits angesprochen, ist für die FoE Bestimmung eine Bewegung notwendig. Während dies bei Flächenmodellen oder Bodenfahrzeugen im Prinzip kein Problem darstellt, schränkt die Verwendung bei Multicopter doch etwas ein. Hierbei wäre evtl. anzudenken für Multicopter statt einem Mono-Kamera-System auf ein Stereo-Kamera-System zu setzen, da bei einem Stereo-Kamera-System eine Bewegung nicht notwendig ist.

Keine Bewegung innerhalb der Szene

In der derzeitigen Umsetzung ist es notwendig, dass sich nur die Drohne bewegt. Dies hat den Grund, dass durch die Bewegung Optische Fluss Vektoren erzeugt werden und dadurch der FoE berechnet wird. Bewegt sich nun aber nicht nur die Drohne, sondern auch z. B. Menschen oder auch andere Objekte innerhalb der Szene, erzeugen diese ebenfalls Optische Fluss Vektoren die jedoch anders gerichtet sind als solche der statischen Szene. Man müsste also für eine Hinderniserkennung in der die Szene nicht statisch ist die Vektoren des Optischen Flusses trennen, um zu unterscheiden, von welchem Objekt diese ausgehen. Sollten jedoch die Objekte weit entfernt sein bzw. nur wenige Objekte, die wenige Vektoren erzeugen, in der Szene sein, stören diese wenig bis gar nicht. Dies liegt daran, dass die Vektoren der Objekte wenig Einfluss auf die FoE Berechnung nehmen und die Ergebnisse nicht verfälschen, da die diese wie Ausreißer behandelt werden. Anders verhält sich dies natürlich, wenn z. B. einer der Würfel sich bewegen würde, da diese doch einen beachtlichen Anteil an der gesamten Szene haben.

Bestimmung der Geschwindigkeit und Höhe

Um die ICT Drohne weiter in Richtung des autonomen Flugs zu entwickeln sollten auch auch weitere Parameter wie z. B. die Höhe und die Fluggeschwindigkeit bestimmt werden. Die derzeit vorhandene Höhenmessung über einen Barometer ist für einen autonomen Flug nicht ausreichend genau. Weiters sollte auch eine Methode gefunden und umgesetzt werden, um die Fluggeschwindigkeit zu messen. Ein GPS Sensor wäre eine Möglichkeit, hat jedoch den Nachteil, dass dieser in Innenräumen nicht funktioniert und im Außenbereich wahrscheinlich zu ungenaue Werte liefert. Eine Möglichkeit, wenn bereits ein Kamerasystem auf der Drohne eingesetzt wird, wäre z. B. Visuelle Odometry. Sollte man ein Mobiltelefon als Plattform verwenden, kann man je nach Gerät die Berechnung der Visuellen Odometry auf einen weiteren CPU Kern auslagern, um so z. B. die Hinderniserkennung sowie Odometry zu vereinen, ohne dass sie sich gegenseitig bei der Verwendung von nur einem CPU Kern stören.

Conclusio

Einige Schwachstellen bzw. Verbesserungsmöglichkeiten der Arbeit wurden in den vorhergehenden Absätzen bereits kurz diskutiert. Der Ansatz einer Hinderniserkennung über den Optischen Fluss hat sich zwar in der Theorie als eine sehr interessante Lösung herausgestellt, welche allerdings in der Praxis einige Schwachstellen aufweist. Die Umsetzung für die Messung der absoluten Entfernung ist durch die angesprochenen Umstände nicht ideal, jedoch wenn es rein um die Erkennung eines Hindernisses geht, kann der Optische Fluss eingesetzt werden. Ein mögliches Szenario ist z. B. wenn sich ein Bodenfahrzeug in einem Raum bewegt. Hier kann mittels Bestimmung der Längen der Vektoren des Optischen Flusses relativ einfach ein Algorithmus zum Auswählen

eines Ausweichmanövers entwickelt werden. Dabei könnte man z. B. das Bild in mehrere Sektoren einteilen, dann die Durchschnittslänge der Vektoren ermitteln und dadurch anschließend bestimmen, in welchem Sektor die Vektoren am längsten sind. Von den längsten Vektoren bewegt man sich weg, da diese signalisieren, dass die Entfernung zu einem Hindernis in diesem Sektor am geringsten ist.

Multicopter werden in den kommenden Jahren noch sehr viel an Forschung und Entwicklung erfahren. Dies hat mitunter den Grund, dass Firmen wie beispielsweise Amazon kommerzielle Interessen verfolgen, aber auch die vielfältigen Möglichkeiten, die Quadrocopter in der Forschung bieten, haben Anteil daran. Während Multicopter derzeit bereits mit diversen Autopiloten GPS Wegpunkte abfliegen können, müssen noch gute Methoden für die Hinderniserkennung gefunden werden. Der Ansatz in dieser Arbeit geht bereits in diese Richtung, konnte aber leider auf Grund der angesprochen Probleme und Schwächen nicht komplett wie geplant umgesetzt werden. Es wurden jedoch Stärken und Schwächen des Optischen Flusses und Ansätze für Verbesserungsmöglichkeiten aufgezeigt. Die Hinderniserkennung ist natürlich nur ein Schritt auf dem Weg hin zu kommerziellen Drohnen. Während es bei der Forschung oder auch im Hobbybereich eher darum geht, den Absturz einer Drohne zu verhindern um den finanziellen Schaden gering zu halten, ist der Anspruch an kommerzielle Drohnen sehr viel höher. Hierbei geht es nicht nur darum, dass die Drohne nicht abstürzt, sondern auch, dass von ihr keine Gefahr für Menschen ausgeht. Auch hierfür ist jedoch wieder eine Hinderniserkennung notwendig. Auch müssen, bis Drohnen wirklich komplett autonom fliegen können, Regulierungsbehörden den Luftraum für Drohnen außerhalb des Sichtbereichs eines Piloten freigeben. Dies ist derzeit noch nicht der Fall, jedoch könnte sich dies durch die rasante Verbreitung von Drohnen, sowie immer besser werdende Technik und nicht zuletzt durch das Lobbying kommerzieller Firmen in den nächsten Jahren ändern.

Abschließend lässt sich sagen, dass für den autonomen Einsatz von Drohnen noch weitere Forschungsarbeit notwendig ist, da vor allem in Hinsicht auf die Verwendung von robusteren und verlässlicheren, sowie genaueren Systemen zur Hinderniserkennung noch Verbesserungsbedarf auf vielen Gebieten besteht.

Wissenschaftliche Literatur

[Ada10] Adamiuk, Grzegorz: *Methoden zur Realisierung von dual-orthogonal, linear polarisierten Antennen für die UWB-Technik*. Bd. 61. KIT Scientific Publishing, 2010

[BCV+11] Bristeau, Pierre-Jean ; Callou, François ; Vissière, David ; Petit, Nicolas [u. a.]: The navigation and control technology inside the ar. drone micro uav. In: *18th IFAC World Congress* Bd. 18, 2011, S. 1477–1484

[BL10] Bredies, K. ; Lorenz, D.: *Mathematische Bildverarbeitung: Einführung in Grundlagen und Moderne Theorie*. Vieweg Verlag, Friedr, & Sohn Verlagsgesellschaft mbH, 2010. – ISBN 9783834898142

[Bou01] Bouguet, Jean-Yves: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. In: *Intel Corporation* 2 (2001), S. 3

[Bra08] Bradski, Adrian: *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]*. 1. ed. O'Reilly Media, 2008. – Gary Bradski and Adrian Kaehler. – ISBN 0–596–51613–4

[Cam95] Camus, Ted: Calculating time-to-contact using real-time quantized optical flow. In: *National Institute of Standards and Technology NISTIR* 5609 (1995)

[CEBENCP08] Cappelle, Cindy ; El Badaoui El Najjar, M ; Charpillet, François ; Pomorski, Denis: Obstacle detection and localization method based on 3D model: distance validation with Ladar. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* IEEE, 2008, S. 4031–4036

[Coh12] Cohen, Yossi: Cross Plattform Computer Vision Optimaization, 2012

[DSTT00] Dellaert, Frank ; Seitz, Steven M. ; Thorpe, Charles E. ; Thrun, Sebastian: Structure from motion without correspondence. In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* Bd. 2 IEEE, 2000, S. 557–564

[Far03] Farnebäck, Gunnar: Two-frame motion estimation based on polynomial expansion. In: *Image Analysis*. Springer, 2003, S. 363–370

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395

[FFH] Freed, Michael ; Fitzgerald, Will ; Harris, Robert: Intelligent Autonomous Surveillance of Many Targets with few UAVS

[FFP12] Friedrich Fraundorfer, Dominik Honegger Gim Hee Lee Lorenz Meier Petri T. ; Pollefeys, Marc: Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV. In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots*

and Systems (IROS) (2012)

[FLM92] Faugeras, Olivier D. ; Luong, Q-T ; Maybank, Stephen J.: Camera self-calibration: Theory and experiments. In: *Computer Vision?ECCV'92* Springer, 1992, S. 321–334

[GFMP08] Gallup, David ; Frahm, J-M ; Mordohai, Philippos ; Pollefeys, Marc: Variable baseline/resolution stereo. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* IEEE, 2008, S. 1–8

[Gib50] Gibson, James J.: The perception of the visual world. (1950)

[GLSU13] Geiger, Andreas ; Lenz, Philip ; Stiller, Christoph ; Urtasun, Raquel: Vision meets Robotics: The KITTI Dataset. In: *International Journal of Robotics Research (IJRR)* (2013)

[GMM12] Gageik, Nils ; Müller, Thilo ; Montenegro, Sergio: Obstacle Detection And Collision Avoidance Using Ultrasonic Distance Sensors For An Autonomous Quadrocopter. In: *University Of Wurzburg, Aerospace Information Technology (Germany) Wurzburg September* (2012)

[GMS+08] Geske, Jon ; MacDougal, Michael ; Stahl, Ron ; Wagener, Jeffrey ; Snyder, Donald R.: Miniature laser rangefinders and laser altimeters. In: *Avionics, Fiber-Optics and Photonics Technology Conference, 2008 IEEE* IEEE, 2008, S. 53–54

[HAC09] Hokuyo Automatic CO., LTD: Scanning Laser Range Finder UTM-30LX/LN Specification. (2009)

[Har94] Hartley, Richard I.: Self-calibration from multiple views with a rotating camera. In: *Computer Vision?ECCV'94*. Springer, 1994, S. 471–478

[HBH+11] Huang, Albert S. ; Bachrach, Abraham ; Henry, Peter ; Krainin, Michael ; Maturana, Daniel ; Fox, Dieter ; Roy, Nicholas: Visual odometry and mapping for autonomous flight using an RGB-D camera. In: *International Symposium on Robotics Research (ISRR)*, 2011, S. 1–16

[HRHM08] Herisse, Bruno ; Russotto, F-X ; Hamel, Tarek ; Mahony, Robert: Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* IEEE, 2008, S. 801–806

[HS81] Horn, Berthold K. ; Schunck, Brian G.: Determining optical flow. In: *1981 Technical Symposium East* International Society for Optics and Photonics, 1981, S. 319–331

[HS97] Hartley, Richard I. ; Sturm, Peter: Triangulation. In: *Computer vision and image understanding* 68 (1997), Nr. 2, S. 146–157

[HZ00] Hartley, R. I. ; Zisserman, A.: *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000

[HZ04] Hartley, R. I. ; Zisserman, A.: *Multiple View Geometry in Computer Vision.* Second. Cambridge University Press, ISBN: 0521540518, 2004

[Inc13] Inc., Qualcomm T.: FastCV SDK Release Notes. 2013

[Ins13] Instruments, Texas: OMAP3530 and OMAP3525 Applications Processors (Rev. H). (2013)

[JC11] Joseph Coombs, Rahul P.: OpenCV on TIs DSP+ARM platforms: Mitigating the challenges of porting OpenCV to embedded platforms, 2011

[JMM05] Johnson, Andrew ; Montgomery, James ; Matthies, Larry: Vision guided landing of an autonomous helicopter in hazardous terrain. In: *Robotics and automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE international conference on* IEEE, 2005, S. 3966–3971

[Kan88] Kanatani, Ken-ichi: Transformation of optical flow by camera rotation. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 10 (1988), Nr. 2, S. 131–143

[Kho11] Khoshelham, Kourosh: Accuracy analysis of kinect depth data. In: *ISPRS workshop laser scanning* Bd. 38, 2011, S. W12

[Kra12] Krause, Stefan: Multi-Purpose Environment Awareness Approach for Single Line Laser Scanner in a Small Rotorcraft UA. In: *Journal of Intelligent & Robotic Systems* 65 (2012), Nr. 1-4, S. 587–601

[LK+81] Lucas, Bruce D. ; Kanade, Takeo [u. a.]: An iterative image registration technique with an application to stereo vision. In: *IJCAI* Bd. 81, 1981, S. 674–679

[LSP08] Lange, Sven ; Sünderhauf, Niko ; Protzel, Peter: Autonomous landing for a multirotor UAV using vision. In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008)*, 2008, S. 482–491

[LW05] Low, Toby ; Wyeth, Gordon: Obstacle detection using optical flow. In: *Proceedings of Australasian Conference on Robotics and Automation 2005* Australian Robotics and Automation Association Inc, 2005

[MBMG01] Merchant, H ; Battaglia-Mayer, A ; Georgopoulos, AP: Effects of optic flow in motor cortex and area 7a. In: *Journal of Neurophysiology* 86 (2001), Nr. 4, S. 1937–1954

[MF92] Maybank, Stephen J. ; Faugeras, Olivier D.: A theory of self-calibration of a moving camera. In: *International Journal of Computer Vision* 8 (1992), Nr. 2, S. 123–151

[MPRPF98] Martinez, JL ; Pozo-Ruz, A ; Pedraza, S ; Fernandez, R: Object following and obstacle avoidance using a laser scanner in the outdoor mobile robot Auriga-α. In: *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on* Bd. 1 IEEE, 1998, S. 204–209

[Mul] Muller, Lupashin Sergei D'Andrea R.: Quadrocopter ball juggling, IEEE. – ISBN 978–1–61284–454–1, S. 5113

[NB] Nikolic, Janosch ; Brescianini, Dario: Ultra Wideband Radar for Micro Aerial Vehicles.

[OK93] Okutomi, Masatoshi ; Kanade, Takeo: A multiple-baseline stereo. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 15 (1993), Nr. 4, S. 353–363

[PH12] Peter Hanger, BSc: Plattform fuer eine autonome Flugdrohne. (2012)

[Pra10] Prabhu, Rahul: C6Accel: TI SoC developers accelerate performance with ready-to-use DSP kernels. (2010)

[PU13] Patel, Dhara ; Upadhyay, Saurabh: Optical flow measurement using lucas kanade method. In: *International Journal of Computer Applications* 61 (2013), Nr. 10, S. 6–10

[RD05] Rosten, Edward ; Drummond, Tom: Fusing points and lines for high performance tracking. In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* Bd. 2 IEEE, 2005, S. 1508–1515

[RD06] Rosten, Edward ; Drummond, Tom: Machine learning for high-speed corner detection. In: *Computer Vision–ECCV 2006*. Springer, 2006, S. 430–443

[SB12] Sebesta, Kenneth ; Baillieul, John: Animal-inspired agile flight using optical flow sensing. In: *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on* IEEE, 2012, S. 3727–3734

[Sch05] Schreer, O.: *Stereoanalyse und Bildsynthese.* Springer, 2005 (SpringerLink: Springer e-Books). – ISBN 9783540234395

[SCW+99] Srinivasan, Mandyam V. ; Chahl, Javaan S. ; Weber, Keven ; Venkatesh, Svetha ; Nagle, Martin G. ; Zhang, Shao-Wu: Robot navigation inspired by principles of insect vision. In: *Robotics and Autonomous Systems* 26 (1999), Nr. 2, S. 203–216

[SHBS11a] Stowers, John ; Hayes, Michael ; Bainbridge-Smith, Andrew: Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. In: *Mechatronics (ICM), 2011 IEEE International Conference on* IEEE, 2011, S. 358–362

[SHBS11b] Stowers, John ; Hayes, Michael ; Bainbridge-Smith, Andrew: Quadrotor Helicopter Flight Control Using Hough Transform and Depth Map from a Microsoft Kinect Sensor. In: *MVA*, 2011, S. 352–356

[sma12] smartmicro: Micro Radar Altimeter Data Sheet. (2012)

[sma13] smartmicro: MRA Type 2 Datasheet. (2013)

[SS99] Shi, Yun Q. ; Sun, Huifang: *Image and video compression for multimedia engineering: fundamentals, algorithms, and standards.* CRC press, 1999

[SZLC96] Srinivasan, M ; Zhang, SW ; Lehrer, M ; Collett, T: Honeybee navigation en route to the goal: visual flight control and odometry. In: *Journal of Experimental Biology* 199 (1996), Nr. 1, S. 237–244

[Tea] Team, OpenEmbedded: OpenEmbedded User Manual

[TGS91] Tistarelli, M ; Grosso, E ; Sandini, G: Dynamic stereo in visual navigation. In: *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on* IEEE, 1991, S. 186–193

[TK95] Taylor, Camillo J. ; Kriegman, David: Structure and motion from line segments in multiple images. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17 (1995), Nr. 11, S. 1021–1032

[Tsa86] Tsai, Roger Y.: An efficient and accurate camera calibration technique for 3D machine vision. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1986*, 1986

[Unb] Unbekannt: SRF02 Ultrasonic range finder ,Technical Specification

[WB95] Welch, Greg ; Bishop, Gary. *An introduction to the Kalman filter.* 1995

[Wil] Willmann, Augugliaro Federico Cadalbert Thomas D'andrea Raffaello Gramazio Fabio Kohler M.: Aerial Robotic Construction Towards a New Field of Architectural Research. In: *International Journal of Architectural Computing, 2012, Vol.10(3), pp.439-460* , S. 439. – ISSN 1478–0771

[Zha00] Zhang, Zhengyou: A flexible new technique for camera calibration. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22 (2000), Nov, Nr. 11, S. 1330–1334. – ISSN 0162–8828

[Zha12] Zhang, Zhengyou: Microsoft kinect sensor and its effect. In: *MultiMedia, IEEE* 19 (2012), Nr. 2, S. 4–10

[ZSS+13] Zhang, Quanshi ; Song, Xuan ; Shao, Xiaowei ; Shibasaki, Ryosuke ; Zhao, Huijing: Category modeling from just a single labeling: Use depth information to guide the learning of 2d models. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* IEEE, 2013, S. 193–200

Internet Referenzen

[1] *Amazon PrimeAir*, 2013. http://www.amazon.com/b?ref_=tsm_1_tw_s_amzn_mx3eqp&node=8037720011.

[2] *Streaming video with Gumstix, GStreamer and the DSP*, September 2013. http://108.163.188.242/~jumpnowt/index.php.

[3] *Using the DSP on Gumstix with Yocto*, September 2013. http://www.sleepyrobot.com/?p=210.

[4] *AR.Drone*, Jannuar 2014. http://ardrone2.parrot.com/.

[5] *Arduino Duemilanove*, Juli 2014. http://arduino.cc/de/Main/ArduinoBoardDuemilanove.

[6] *Barbar Pole Illusion*, September 2014. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html.

[7] *Boeing A160 Hummingbird*, September 2014. http://commons.wikimedia.org/wiki/File:Boeing_A160_Hummingbird_VTOL-UAS.jpg.

[8] *Bumblebee 2 Stereo Kamera*, Februar 2014. http://ww2.ptgrey.com/stereo-vision/bumblebee-2.

[9] *Bumblebee Stereo Kamera*, September 2014. https://flic.kr/p/4YE6PE.

[10] *C6Accel vs Cortex A8*, März 2014. http://processors.wiki.ti.com/index.php/C6EZAccel_FAQ#What_DSP_kernels_can_I_assess_through_C6Accel.

[11] *C6EZAccel*, März 2014. http://processors.wiki.ti.com/index.php/C6EZAccel.

[12] *Centeye Optical Flow*, Februar 2014. http://www.centeye.com/technology/optical-flow/.

[13] *Comparison of the OpenCV feature detection algorithms*, März 2014. http://computer-vision-talks.com/articles/2011-01-04-comparison-of-the-opencv-feature-detection-algorithms/.

[14] *DHL Drohne*, September 2014. http://upload.wikimedia.org/wikipedia/commons/0/0e/Package_copter_microdrones_dhl.jpg.

[15] *Dragon Board*, März 2014. http://mydragonboard.org/.

[16] *Dragonboard*, September 2014. http://mydragonboard.org/db8074/.

[17] *DSP Library Übersicht*, März 2014. http://processors.wiki.ti.com/index.php?title=Software_libraries.

[18] *GNU C Compiler - Options That Control Optimization*, März 2014. http://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html.

[19] *GStreamer - open source multimedia framework*, März 2014. http://gstreamer.freedesktop.org/.

[20] *Gumstix FireSTORM*, März 2014. https://store.gumstix.com/index.php/products/267/.

[21] *Gumstix GIT Repository mit Kurzanleitung*, März 2014. https://github.com/gumstix/Gumstix-YoctoProject-Repo.

[22] *Gumstix Linux Images*, März 2014. http://gumstix.org/download-prebuilt-images.html.

[23] *Gumstix Mailing List*, März 2014. http://sourceforge.net/p/gumstix/mailman/?source=navbar.

[24] *Gumstix SD Card*, März 2014. http://gumstix.org/create-a-bootable-microsd-card.html.

[25] *Gumstix Summit Extension Board*, März 2014. https://store.gumstix.com/index.php/products/215/.

[26] *Hokuyo Laser Scanner*, Februar 2014. http://www.robotshop.com/en/hokuyo-utm-03lx-laser-scanning-rangefinder.html.

[27] *Illustrates how image rectification simplifies the search space in stereo correlation matching.*, September 2014. https://flic.kr/p/aiBdJB.

[28] *Khronos - OpenVX*, März 2014. http://www.khronos.org/openvx/.

[29] *KITTI Vision Benchmark Suite*, März 2014. http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=flow.

[30] *Microsoft Kinect*, September 2014. http://upload.wikimedia.org/wikipedia/commons/6/67/Xbox-360-Kinect-Standalone.png.

[31] *Microsoft Kinect Beispielcode*, Februar 2014. http://msdn.microsoft.com/en-us/library/hh855376.aspx.

[32] *Microsoft Kinect FAQ*, Februar 2014. http://www.microsoft.com/en-us/kinectforwindows/Faq.aspx.

[33] *Microsoft Kinect Launch*, Februar 2014. http://gizmodo.com/5563148/microsoft-xbox-360-kinect-launches-november-4.

[34] *Minoru 3D Webcam*, September 2014. https://flic.kr/p/4YE6PE.

[35] *Minoru 3D Webcam*, Februar 2014. http://www.minoru3d.com/.

[36] *MLR100 Preis*, Februar 2014. http://www.laserscanningforum.com/forum/viewtopic.php?f=93&t=4498.

[37] *Novelda - Robot Vision*, Februar 2014. https://www.novelda.no/content/application-ideas.

[38] *OpenCV Camera Calibration*, Februar 2014. http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[39] *OpenCV DSP Support*, März 2014. http://e2e.ti.com/support/embedded/tirtos/f/355/t/125780.aspx.

[40] *OpenCV DSP Support*, März 2014. https://code.google.com/p/opencv-dsp-acceleration/.

[41] *Origenboard*, März 2014. http://www.origenboard.org/.

[42] *Pandaboard Bild*, September 2014. https://flic.kr/p/dPaSRt.

[43] *Pandaboard Website*, September 2014. http://pandaboard.org/.

[44] *Point Cloud Library*, September 2014. https://flic.kr/p/4YE6PE.

[45] *Point Cloud Library (PCL)*, Februar 2014. http://www.pointclouds.org/.

[46] *Project Tango*, Juli 2014. https://www.google.com/atap/projecttango/#project.

[47] *Servo Library*, Juli 2014. http://arduino.cc/de/Reference/Servo.

[48] *smartmicro - Airborne Radar Overview*, Februar 2014. http://www.smartmicro.de/index.php/en/airborne-radar/airborne-radar-overview.

[49] *SRF02*, September 2014. https://flic.kr/p/bDWsaR.
[50] *SRF02 Ultraschall Sensor*, Februar 2014. http://de.manu-systems.com/SRF02.shtml.
[51] *Syma S107 Helikopter*, September 2014. https://flic.kr/p/aiBdJB.
[52] *T Rex 450 se v2*, September 2014. https://flic.kr/p/HjKCP.
[53] *Tegra Android Development Pack*, März 2014. http://docs.nvidia.com/gameworks/index.html#technologies/mobile/opencv_intro.htm.
[54] *XDAIS*, März 2014. http://processors.wiki.ti.com/index.php/Category:XDAIS.
[55] futurezone.at. *Bundesheer-Drohnen erkennen viel, aber nicht alles*, November 2013. http://futurezone.at/digital-life/bundesheer-drohnen-erkennen-viel-aber-nicht-alles/33.540.297.
[56] Tages Anzeiger. *Studenten bauen Mini-Helikopter für Lawinenopfer*, Mai 2010. http://www.tagesanzeiger.ch/wissen/technik/Studenten-bauen-MiniHelikopter-fr-Lawinenopfer/story/22710938.

A Anhang

A.1 Gumstix Overo - Bitbake Rezept

```
DESCRIPTION = "A basic console image for Gumstix boards."
LICENSE = "MIT"
PR = "r0"

IMAGE_FEATURES += "splash package-management ssh-server-openssh tools-sdk"
#IMAGE_LINGUAS = "en-us"
IMAGE_LINGUAS = "de-de"

inherit core-image

# Gumstix machines individually RDEPEND on the firware they need but we repeat
# it here as we might want to use the same image on multiple different machines.
FIRMWARE_INSTALL = " \
  linux-firmware-sd8686 \
  linux-firmware-sd8787 \
"

TOOLS_INSTALL = " \
  alsa-utils \
  e2eaudiotest \
  systemd-analyze \
  cpufrequtils \
  grep \
  gzip \
  iputils \
  iw \
  memtester \
  nano \
  sudo \
  tar \
  tslib \
  u-boot-mkimage \
  u-boot-fw-utils \
  vim \
  wget \
  zip \
```

```
  media-ctl \
  yavta \
  v4l-utils \
  mplayer2 \
  iperf \
  raw2rgbpnm \
  x11vnc \
"

# TI and OpenCV
TI_OpenCV_INSTALL = " \
 gstreamer-ti \
 gst-plugins-good-meta \
 gst-plugins-base-meta \
 gst-plugins-bad-meta \
 ti-dmai \
 opencv \
 opencv-dev \
 opencv-samples \
 binutils \
 binutils-symlinks \
 cpp \
 cpp-symlinks \
 diffutils \
 file \
 gcc \
 gcc-symlinks \
 g++ \
 g++-symlinks \
 gettext \
 ldd \
 libstdc++ \
 libstdc++-dev \
 libtool \
 make \
 pkgconfig \
"

IMAGE_INSTALL += " \
  packagegroup-cli-tools \
  packagegroup-cli-tools-debug \
  ${FIRMWARE_INSTALL} \
  ${TOOLS_INSTALL} \
  ${TI_OpenCV_INSTALL} \
"

add_custom_smart_config() {
        smart --data-dir=${IMAGE_ROOTFS}/var/lib/smart channel --add gumstix
            type=rpm-md name="Gumstix Package Repository"
            baseurl=http://package-cache.s3-website-us-west-2.amazonaws.com/dev/ -y
}
set_gumstix_user() {
        echo "gumstix:x:500:" >> ${IMAGE_ROOTFS}/etc/group
```

```
        echo "gumstix:VQ43An5F8LYqc:500:500:Gumstix User,,,:/home/gumstix:/bin/bash" >>
            ${IMAGE_ROOTFS}/etc/passwd

        install -d ${IMAGE_ROOTFS}/home/gumstix
        cp -f ${IMAGE_ROOTFS}/etc/skel/.bashrc ${IMAGE_ROOTFS}/etc/skel/.profile
            ${IMAGE_ROOTFS}/home/gumstix
        chown gumstix:gumstix -R ${IMAGE_ROOTFS}/home/gumstix

        echo "%gumstix ALL=(ALL) ALL" >> ${IMAGE_ROOTFS}/etc/sudoers
        chmod 0440 ${IMAGE_ROOTFS}/etc/sudoers
        chmod u+s ${IMAGE_ROOTFS}/usr/bin/sudo
}

ROOTFS_POSTPROCESS_COMMAND =+ "set_gumstix_user ; add_custom_smart_config ;"
```

Code A.1: Bitbake Rezept zum erstellen eines Images für den Gumstix Overo inkl. OpenCV und TI DSP unterstützung.