Masterarbeit am Institut für Informatik der Freien Universität Berlin,

Biorobotics Lab der Arbeitsgruppe Intelligente Systeme und Robotik

Kollisionserkennung für Single-/Multicopter mit Hilfe des Optischen Flusses

Eric Zetzsche
Matrikelnummer: 4287592
ericz@inf.fu-berlin.de

Betreuer: Dr. Tim Landgraf
Zweitgutachter: Prof. Dr. Raul Rojas

Berlin, 25.08.2014

Zusammenfassung

In dieser Arbeit wird ein Algorithmus vorgestellt, der es ermöglicht, anhand des Optischen Flusses Kollisionen erkennen zu können. Sie ist eingebettet in den Rahmen des Neurocopter-Projekts. Der Ansatz beruht auf dem Berechnen von durchschnittlichen Flüssen der in Teilbilder unterteilten Bildern. Diese Bilder werden von der an der Frontseite eines Quadrocopters angebrachten Kamera aufgenommen und von dem mitfliegenden Mini-PC verarbeitet. Die Einschätzung der Situationen erfolgt per Mustervergleich, mithilfe von multivariaten Normalverteilungen. Dabei wurden gute Ergebnisse erreicht, sofern sich die Bewegungen auf Translationen beschränken. Drehungen führen häufig zu falsch erkannten Kollisionen. Zudem führt der größere Winkel bei zunehmender Geschwindigkeit zu Problemen. Es gibt allerdings Möglichkeiten, diese Probleme zu lösen. Diese sollten in zukünftigen Arbeiten getestet werden.

Danksagung

An dieser Stelle möchte ich allen danken, die mich in der Zeit der Erstellung dieser Arbeit unterstützt haben. Mein besonderer Dank gilt dabei Dr. Tim Landgraf, der mich betreut und diese Arbeit überhaupt erst ermöglicht hat. Natürlich möchte ich nicht Simon Wichmann vergessen, der meine Arbeit fachlich Korrektur gelesen hat. Zudem Kaja Brinkmann, die für einen Rohkostkuchen diese Arbeit orthographisch und grammatikalisch verbessert hat. Ein großes Danke geht natürlich auch an meine Freundin Saskia Göbel, die mich während der Zeit nach besten Mitteln unterstützt hat, und natürlich meinen Eltern, ohne deren Hilfe dies alles nicht möglich gewesen wäre.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

25.08.2014

Eric Zetzsche

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Arbeit entstand im Rahmen des Neurocopter-Projekts des Biorobotic Labs, einer Untergruppe der AG Intelligente Systeme und Robotik der FU Berlin. Das primäre Ziel der Gruppe ist ein besseres Verständnis der Natur durch ihre Erforschung mithilfe von Robotik, z.B. durch Imitation von natürlichen Verhaltensweisen oder durch Integration eines Roboters in einen spezifischen Lebensraum. In diesem Rahmen besteht u.a. die Erwartung, weitere biologische Erkenntnisse zu erhalten sowie davon inspirierte Technologien zu erlangen.

Das Neurocopter-Projekt hat das Ziel, einen autonomen Quadrocopter zu entwickeln, der das Verhalten und die Vorgehensweisen von Honigbienen imitiert, um so bestehende Theorien betreffend der Orientierung und Navigation von Bienen zu bestätigen oder zu widerlegen.

Um die Verhaltensweisen der Honigbiene möglichst authentisch zu imitieren, soll dem Copter die gleiche Sensorik zur Verfügung stehen wie auch der Biene. Zudem sollte der Neurocopter günstig herstellbar beziehungsweise nachbaubar sein. Somit kommen für den Copter nur handelsübliche, gebräuchliche Bauteile in Frage. Einzige Ausnahme bildet der geplante Neuromorphe Chip, der zum Zeitpunkt dieser Arbeit noch in der Entwicklung ist. Dies ist eine spezialisierte Prozessoreinheit, die für die Berechnung von neuronalen Netzen entworfen wurde. Geplant ist dafür eine Zusammenarbeit mit Jun. Prof. Dr. Elizabeth Chicca [NBR+13] der Universität Bielefeld oder dem Ko-Direktor des Human Brain Projects [Pro], Prof. Dr. Karlheinz Meier [Hei], von der Universität Heidelberg. Beide arbeiten getrennt an der Entwicklung von Neuromorphen Chips. Diese Prozessoren hätten die nötigen Voraussetzungen, um in Echtzeit ein Bienengehirn zu simulieren, was mit herkömmlichen Prozessoren nicht möglich wäre. Berechnungen für solch große neuronale Netze sind sehr aufwendig und sind im Moment auf handelsüblichen Mini-Computern nicht zu bewerkstelligen. Das neuronale Netzwerk soll so weit wie möglich ein echtes Bienengehirn simulieren, daher wird ein besonderes neuronales Netz verwendet, ein *Spiking Neural Network*. Dieses Netzwerk soll am Ende den Copter steuern. Dafür soll es lernen, sich zu orientieren und zu navigieren wie eine Biene.

In diesem Rahmen wird auch eine Kollisionsvermeidung benötigt, welche in dieser Arbeit versucht wird zu entwickeln. Dabei steht nur der Optische Fluss zur Verfügung. Erschwerend kommt in diesem Zusammenhang die begrenzte Rechenleistung hinzu, da sowohl der Platz als auch das Gewicht beim Neurocopter stark limitiert sind.

Ein positiver Aspekt der Arbeit ist die mögliche Reduzierung von Senso-

ren, da der Optische Fluss für andere Funktionen obligatorisch ist und man sich so auf Daten stützen kann, die bereits vorliegen. Weiterhin ist das Ziel, eine Kollisionserkennung zu entwickeln, die kosteneffizient, leichtgewichtig und einfach nachzubauen ist. Eine günstige und effiziente Kollisionserkennung kann auch außerhalb des Copters nützlich sein. Zum Beispiel könnten sehbehinderte Menschen von solchen Systemen profitieren, indem es ihnen bessere Möglichkeiten zur Orientierung bietet. Aber damit endet die Einsatzmöglichkeit eines solchen Systems nicht, auch diverse Fahrzeuge könnten damit ausgerüstet werden, um die Sicherheit im Straßenverkehr zu erhöhen.

Als Methode für die Kollisionserkennung dient das Supervised Learning, durch das versucht wird, anhand von geeigneten Daten potentiell gefährliche Situationen zu erkennen. Dafür soll eine möglichst große Anzahl an Flugvideos erstellt werden, in denen die Situationen repräsentiert werden. Nach einer Vorverarbeitung dieser Daten werden sie verwendet, um Entscheidungen treffen zu können, ob eine Situation zu einer Kollision führen kann oder nicht.

2 Verwandte Arbeiten

Es gibt bereits einige andere Projekte und Arbeiten mit dem Ziel der Kollisionserkennung. Alle verwenden hauptsächlich nur den Optischen Fluss, um das Ziel zu erreichen, allerdings mit anderen Ansätzen als in dieser Arbeit.

Sebesta und Baillieul entwickeln in ihrer Arbeit[SB12] eine Methode, um mithilfe des Optischen Flusses, inspiriert durch den Vogelflug, autonome Drohnen durch ein Gebiet von Hindernissen fliegen zu lassen. Dabei verwenden sie die *Time-To-Contact* (siehe Kapitel 9), berechnet anhand der Brennweite der nach vorn ausgerichteten Kamera und der Geschwindigkeit. Als Plattform kommt ebenfalls ein Quadcopter in ihrer Arbeit zum Einsatz.

Einen ähnlichen Ansatz verwenden auch Souhila und Karim in ihrer Arbeit [SK07]. Sie verwenden den Fluchtpunkt und auch die *Time-To-Contact*, um einen autonomen mobilen Roboter Hindernisse erkennen und vermeiden zu lassen. Sie verwenden dabei Kontrollregeln, die sie „balance strategy“ nennen. Es wird dabei versucht, den Fluss auf der rechten und linken Seite des Focus-of-Expansion gleich groß zu halten. Sie machen sich so die Eigenschaft zunutze, dass näher gelegene Objekte einen relativ größeren Optischen Fluss erzeugen als entferntere Objekte. Ob sich die Arbeit auf eine fliegende Drohne übertragen lässt, ist ohne weiteres schwer abzuschätzen. Vermutlich werden die hinzukommenden Freiheitsgrade ein größeres Problem für diesen Ansatz darstellen.

Eine dritte Arbeit von Green und Oh [GO08] versucht das Verhalten von fliegenden Insekten zu imitieren. Sie verwenden Starrflügler mit zwei Kameras, die im 45°-Winkel von der Flugrichtung links und rechts angebracht sind. Aus dem Optischen Fluss werden für jede Seite Winkelgeschwindigkeiten von Objekten im Sichtfeld berechnet. Überschreitet die Winkelgeschwindigkeit eines Objektes einen Schwellwert, so dreht sich die Drohne von dieser Richtung weg. Ein großes Problem dieses Ansatzes stellen schmale sowie große Objekte dar. Diese werden häufig nicht als Hindernis erkannt, da schmale Objekte unter Umständen zwischen den Sichtfeldern der Kameras liegen und große Objekte wie Wände oft sehr homogene Strukturen besitzen, die eine Berechnung des Optischen Fluss unmöglich machen können (siehe Abbildung 1).
Ähnliche Ziele des Biorobotic Labs verfolgt Srinivasan in seinen Arbeiten. Unter anderem untersuchte er in seiner Arbeit die Navigations- und Orientierungsmethoden von Honigbienen [Sri11], um sie in der Robotertechnik verwenden zu können. Er verwendet den Optischen Fluss unter anderem um Drohnen autonom landen und selbständig durch Korridore fliegen zu lassen. Alle Arbeiten erzielten recht gute Ergebnisse mit ihren Ansätzen, jedoch zielt keine bis auf die von Sebesta und Baillieul auf eine Nutzung in einem

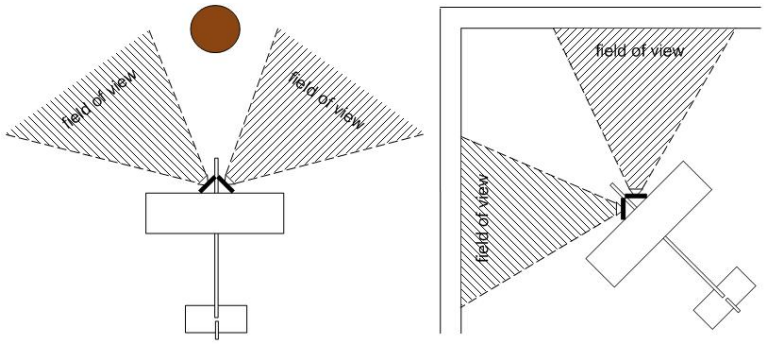

Abbildung 1: Greens und Ohs Ansatz hat starke Probleme mit schmalen Objekten und großen Objekten.

Multicopter ab. Solch eine Plattform besitzt allerdings Eigenschaften, die in diesen Arbeiten nicht berücksichtigt wurden, wie zum Beispiel die unterschiedlichen Neigungswinkel bei unterschiedlichen Geschwindigkeiten eines Copters. Selbst Sebesta und Baillieul haben nicht alle Aspekte betrachtet. In ihrer Arbeit wurde unter anderem der Winkel des Quadrocopters im Flug nicht beachtet, zudem wird von einer Bewegung parallel zum Boden ausgegangen. Diese Arbeit hat das Ziel, mit einem anderen Ansatz, spezialisiert auf Quadrocopter, ein besseres Ergebnis zu erreichen.

3 Verwendete Methoden

In diesem Kapitel werde ich auf die verwendeten Methoden und Algorithmen eingehen. Zur Berechnung des Optischen Flusses werden die Algorithmen von Shi-Tomasi und Lucas-Kanade verwendet. Weiterhin nutze ich die Hauptkomponentenanalyse zur Reduzierung der Dimensionalität, um die Effizienz und Verallgemeinerung der Ground Truth-Daten zu steigern.
Als Ground Truth-Daten werden in dieser Arbeit Daten bezeichnet, die zum Lernen des Algorithmus verwendet werden. Es ist von Nutzen, diese Algorithmen zu verstehen, bevor wir zur eigentlichen Arbeit kommen.

3.1 Harris & Shi-Tomasi-Eckenerkennung

Es werden prägnante Punkte in Bildern benötigt, um sie in darauffolgenden Bildern wiedererkennen zu können. In dieser Arbeit wird die Shi-Tomasi-Eckenerkennung [ST93] verwendet, welche auf der Harris-Eckenerkennung [HS88] basiert. Beide versuchen den Punkt mit dem höchsten Intensitätsunterschied in alle Richtungen zu finden. Mathematisch lässt sich der Unterschied in alle Richtungen (*E(u,v)*) eines Punktes mit den Koordinaten u und v wie folgt beschreiben:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Wobei w(x,y) die *Fensterfunktion* denotiert, welche nicht weiter definiert ist. Es kann zum Beispiel ein Rechteck-Fenster, ein Gauß-Fenster oder ähnliches sein. Diese Funktion gilt es zu maximieren, um den prägnantesten Punkt zu finden. Dafür wird sie mithilfe der Taylorreihe umgeformt. Daraus folgt die Annäherung:

$$E(u,v) \approx [u\ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

Dabei ist M wie folgt definiert:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

Wobei I_x und I_y die partiellen Ableitungen nach x und y sind.

Um diese Annäherung zu maximieren, muss man sich die Matrix M ansehen. Damit man den maximalen Unterschiedswert ($E(u,v)$) erhält und somit eine Ecke oder ein markanter Punkt vorliegt, müssen die Werte der Matrix möglichst groß sein. Dies erreicht man mithilfe der Varianz. Man kann erkennen, dass die Matrix

$$\begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

eine Kovarianzmatrix über x und y darstellt. Deren Werte sind also groß, wenn auch die Varianzen und die Kovarianzen groß sind. Wie auch in Kapitel 3.3 beschrieben, bezeichnen die Eigenwerte einer Kovarianzmatrix die Varianzen der zugrunde liegenden Daten. Man kann also die Eigenwerte verwenden, um die Größe von $E(u, v)$ einzuschätzen. Aus dieser Erkenntnis wird ein Score R berechnet, der verwendet wird, um einzuschätzen, ob ein Punkt eine Ecke oder ein markanter Punkt ist. Harris' Gleichung für den Score lautet wie folgt:

$$R = det(M) - k(trace(M))^2$$

Darauf aufbauend stellen Shi und Tomasi für R folgende Gleichung auf:

$$R = min(\lambda_1, \lambda_2)$$

Übersteigt dieser Wert einen vorher festgelegten Schwellwert, wird er als markanter Punkt angenommen. Oft wird der Schwellwert anhand des besten Scores ermittelt. Ist zum Beispiel der höchste Score 100 und man hat einen Faktor von 0.1 festgelegt, dann erhält man einen Schwellwert von $100 * 0.1 = 10$ für den Score. Der Faktor ist ein Parameter und kann den Gegebenheiten entsprechend gewählt werden.

3.2 Lucas-Kanade-Methode

Um den Optischen Fluss zu bestimmen, werden vorher bestimmte Punkte bzw. Ecken versucht wiederzuerkennen. Lucas und Kanade[LK81] verwenden dabei einen ähnlichen Ansatz wie Shi und Tomasi für die Eckenerkennung. Um die Gleichung des Optischen Flusses, erstmals aufgestellt von Horn und Schunck [HS81] zu lösen, genügt ein einzelner Punkt nicht aus, da so die Gleichung unterbestimmt wäre. Um dem Problem entgegenzuwirken, verwenden sie einen $m \times m$ Ausschnitt um ein Feature herum. Sie gehen davon aus, dass alle Punkte in diesem Ausschnitt einen gleichen Optischen Fluss besitzen, also die gleiche Bewegung vollziehen. Im Fall der OpenCV-Implementierung kommt standardmäßig ein 3×3 Feld zum Einsatz. Die Größe kann aber als Parameter gesetzt werden.
Die Funktion zur Berechnung des Optischen Flusses lautet im zweidimensionalen Raum wie folgt:

$$\begin{aligned} I_{x_1} V_x + I_{y_1} V_y &= -I_{t_1} \\ I_{x_2} V_x + I_{y_2} V_y &= -I_{t_2} \\ &\vdots \\ I_{x_n} V_x + I_{y_n} V_y &= -I_{t_n} \end{aligned}$$

Dabei sind I_x, I_y, I_t die partiellen Ableitungen des Bildes nach x, y und der Zeit und n die Anzahl der Pixel im Bildausschnitt. V_y und V_x bezeichnen den Optischen Fluss in x- und y-Richtung. Da das Gleichungssystem mit 9 Punkten (bzw. $m \times m$ Punkten) überbestimmt ist, kann man es mit der Methode der kleinsten Quadrate lösen und erhält:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i I_{x_i}^2 & \sum_i I_{x_i} I_{y_i} \\ \sum_i I_{x_i} I_{y_i} & \sum_i I_{x_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_{x_i} I_{t_i} \\ -\sum_i I_{y_i} I_{t_i} \end{bmatrix}$$

Man erkennt deutlich die Ähnlichkeit zur Shi-Tomasi-Gleichung.

3.3 Hauptkomponentenanalyse (PCA)

Manchmal ist es sinnvoll, die Dimensionalität einer Aufgabe zu minimieren, um zum Beispiel die Effizienz eines Algorithmus zu erhöhen oder diesen zu verallgemeinern. Reduziert man zum Beispiel die Dimensionalität einer Gleichung, benötigt man weniger Punkte, um diese zu lösen. Hierfür kann man die Hauptkomponentenanalyse, entwickelt von Karl Pearson[Pea01], verwenden.
Die PCA wird auf eine Punktwolke angewendet, über die man die paarweisen Abhängigkeiten der Dimensionen minimieren möchte. Es wird dabei eine Transformation der Achsen vorgenommen, um eine möglichst hohe Unabhängigkeit der einzelnen Dimensionen zueinander zu schaffen. Das bedeutet, man strebt eine Minimierung der Abstände von Punkten zu jeder Achse an, sodass bei einer Projektion auf eine Achse der Fehler möglichst klein ist. Dies kann erreicht werden durch die Maximierung der Varianzen. Um dies besser zu veranschaulichen, ist ein Beispiel in Abbildung 2 dargestellt. Hier sieht man, dass, wenn man die schwarze Linie um den Nullpunkt rotiert, die Varianz am größten ist, wenn die Fehler minimal sind.

Ein iteratives Herangehen wäre hier zu aufwendig. Es kann gezeigt werden, dass die Eigenvektoren der Kovarianzmatrix die Linearkombinationen der Achsen darstellen, die die Varianzen maximieren. Sie stellen also die Hauptkomponenten dar. Ebenso kann gezeigt werden, dass die Eigenwerte die Größen der Varianzen darstellen. Zum Beispiel hat die Hauptkomponente, die aus der Linearkombination des ersten Eigenvektors gebildet wird, eine Varianz, die so groß ist wie der erste Eigenwert. So genügt es, die Eigenwerte und Eigenvektoren zu berechnen. Die Hauptkomponenten können anhand der Varianzen sortiert werden.

3.4 Konvolution

In dieser Arbeit verwende ich die Konvolution, um eine Glättung der Daten vorzunehmen. Dabei wird für jeden Punkt dessen gewichteter Mittelpunkt

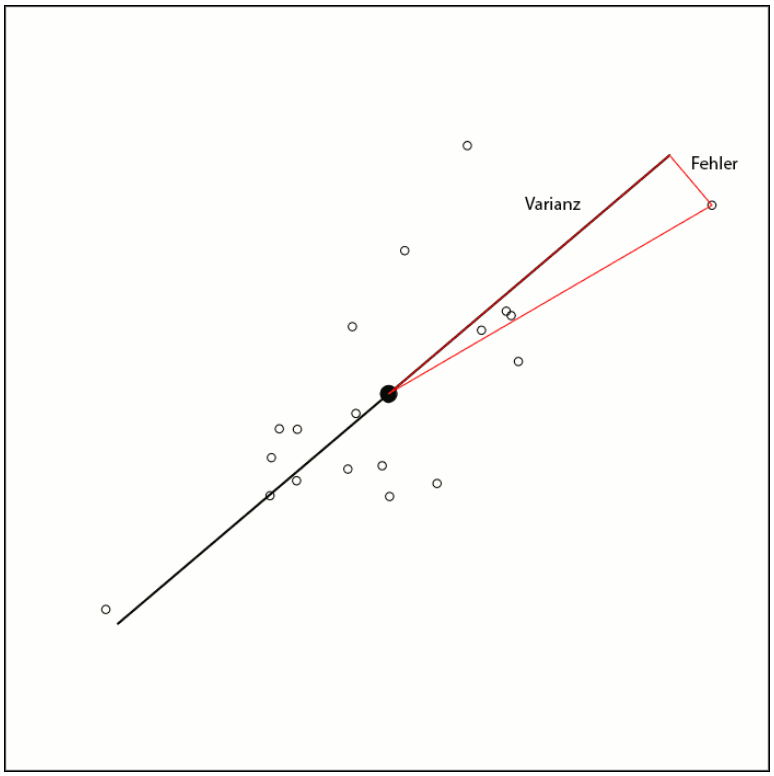

Abbildung 2: PCA 2D Beispiel: Es wird versucht eine Achse (schwarze Linie) so zu legen, dass die Varianz der Daten, projiziert auf die Achse, maximal und somit die Fehler minimal sind.

errechnet. Dafür wird ein Fenster über die Daten geschoben und der gewichtete Mittelwert für jede Position des Fensters berechnet. Um die Punkte im Fenster zu gewichten, wird ein sogenannter Kernel verwendet. Mathematisch lässt sich dies wie folgt darstellen:

$$f(x_i) = \frac{x_{i-n+1} * k_0 + x_{i-n+2} * k_1 + \ldots + x_i * k_n}{n}$$

Wobei x_i den i-ten Punkt denotiert und k_i den i-ten Wert des Kernels. Den Kernel kann man als eine Fensterfunktion ansehen, die über die Daten geschoben wird.

4 Der Neurocopter

Meine Arbeit ist Teil des Neurocopter-Projekts und hat daher entsprechende Vorgaben. Der Neurocopter (Abbildung 3), der die Plattform dieser Arbeit darstellt, ist ein selbstgebauter Quadrocopter, speziell konstruiert für die Anforderungen des Projekts. Es folgt eine Beschreibung der Gegebenheiten.

Der Copter besteht aus einem Karbon-Chassis mit vier Motoren, einem Arduino Chipsatz [sto], einem Odroid Linux-Board [co.] und den Sensoren. Das Arduino und das Odroid stellen die Steuereinheiten des Copters dar und teilen sich die dazugehörenden Aufgaben.

Das Arduino ist ein spezialisiertes Arduino Board, das ArduPilot Mega 2.6, welches bereits einige wichtige Sensoren besitzt, die für Drohnen wichtig sind. Dazu gehören ein 3-Achsen-Gyroskop, ein Accelerometer und ein Luftdrucksensor. Es verarbeitet Navigationsbefehle, die entweder im manuellen Flug von der Fernsteuerung oder im autonomen Flug vom Odroid/Neuromophem Chip gesendet werden.
Das ArduPilot verwendet die ihm zur Verfügung stehende Sensorik, um die Navigationsbefehle in Steuerungsbefehle für die Motoren bzw. Motorensteuerungen umzusetzen. Wir verwenden ein dafür entwickeltes Open Source Framework, das ArduCopter Framework. Es fungiert als eine Art low-level Flugcomputer. Mit dem Begriff low-level ist in diesem Fall eine niedrige Abstraktionsebende gemeint. Das Arduino arbeitet also nah an den physikalischen Komponenten des Copters.

Der Odroid ist ein recht günstiger Linux Computer mit einem 1.7 GHz Quad-Core Prozessor und 2 GByte RAM. Das Betriebssystem ist ein minimales Linux, um Ressourcen zu sparen. Darauf läuft eine Software mit einem in der Arbeitsgruppe entwickelten Framework, dem Berlin United-Framework [UF], welches high-level Aufgaben übernimmt, wie die Kollisionskontrolle oder die Weiterleitung der Nachrichten vom Arduino an die Bodenstation. Die Bodenstation dient hierbei lediglich zur Überwachung und maximal zur Zielsetzung.

Die zur Verfügung stehenden Sensoren sind eine Frontkamera, ein Gyroskop, ein Kompass, GPS, Barometer und ein Sonar. Die Frontkamera, eine Playstation Eye, liefert die Bilder in der Auflösung 640 x 480 Pixel bei einer Bildrate von maximal 75 Bildern pro Sekunde.

Abbildung 3: Der Neurocopter, von der Gruppe des gleichnamigen Projektes entworfen und gebaut. Er stellt die Plattform dieser Arbeit dar.

5 Implementierung

Die Implementierung erforderte mehrere unabhängige Bausteine, welche je nach Anforderung auch in unterschiedlichen Sprachen entwickelt wurden. Dies ließ sich recht gut bewerkstelligen, da sich die Methoden der Arbeit gut in einzelne voneinander unabhängige Phasen unterteilen lassen. Zur Verwendung kamen hier:

- die Skriptsprache R: für Statistik
- Java 1.7: für alle kleineren Tools, wie zum Beispiel das Umwandeln von Dateiformaten
- C++11: für alle auf dem Odroid laufenden Programme, wie auch die Kollisionserkennung

Es wird auf dem Odroid das NeurocopterOnBoard-Programm verwendet, welches das Berlin United Framework nutzt. Das Programm verwaltet alle Module und bietet diverse Schnittstellen zur FURemote Software (siehe Abbildung 4). Diese dient zur Überwachung und zum besseren Debugging. Durch Modularisierung ermöglicht das Framework ebenso eine leichtere Erweiterung. Die im Rahmen des Neurocopter entwickelten Module werden durch verschiedene Scheduler des Frameworks ausgeführt. Es gibt Module, die Round Robin-artig, zyklisch ausgeführt werden, und Module, die abhängig von Daten ausgeführt werden. Die Kollisionskontrolle, die in dieser Arbeit entwickelt wurde, gehört zum zweiten Fall und wird immer dann ausgeführt, wenn die Kamera ein neues Bild zur Verfügung stellt.

Die Software FURemote dient als eine Art Bodenstation und ist in der Lage, vom NeurocopterOnBoard-Programm gesendete Informationen anzuzeigen

und darzustellen. Dazu gehören Bilder und Grafiken sowie einfache Variablenwerte. Sie ist in der Sprache Java mit der Erweiterung des Eclipse RCP Frameworks entwickelt worden.

FURemote sowie das Berlin United Framework wurden ursprünglich für das FUmanoids Projekt der AG Künstliche Intelligenz der FU Berlin entwickelt.

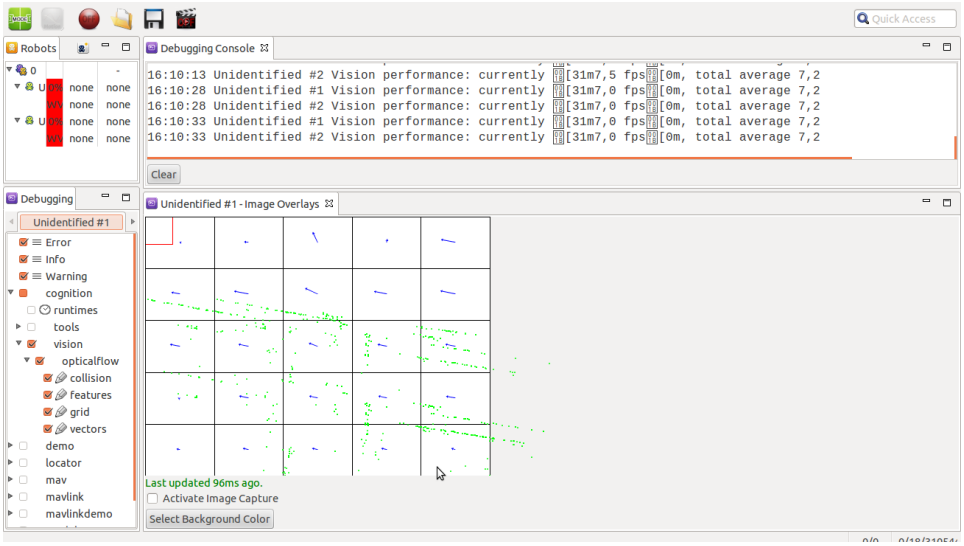

Abbildung 4: FURemote ist in der Lage Variablen Werte und Grafiken zur Laufzeit darzustellen und ermöglicht so ein leichteres Debugging. Hier ist die Anzeige des momentanen Optischen Flusses zu sehen. Die grünen Punkte stellen die Features dar und die blauen Pfeile die durchschnittlichen Flüsse.

5.1 Phasen

Die Herangehensweise lässt sich in mehrere Phasen bzw. Schritte unterteilen.

Vorbereitung:

1. Aufnahme der kategorisierten Daten
2. Einteilung der Einzelbilder in Zellen
3. Berechnung des durchschnittlichen Flusses für jede Zelle
4. Glättung der Flüsse
5. Berechnung der Hauptkomponenten
6. Erstellung einer multivariaten Normalverteilung für jede Kategorie möglicher Situationen

7. Ermitteln von Schwellwerten

Während des Fluges:

Während des Fluges werden folgende Schritte für jedes Bild durchgeführt.

1. Einteilung jedes Einzelbildes der Frontkamera in Zellen
2. Berechnung des durchschnittlichen Flusses für jede Zelle
3. Glättung der Flüsse
4. Umformung der Flüsse auf die Hauptachsen
5. Berechnung der Dichten für jede Normalverteilung
6. Entscheidung der Gefährlichkeit der Situation

5.2 Die Aufnahme der Daten

Dies ist ein Thema, von dem ich zu Beginn der Arbeit keine Probleme erwartet hatte. In der Praxis gestaltete sich das Sammeln der Daten als weitaus aufwendiger als gedacht. In ihrer Grundform sind die Daten Videos, die während des Fluges von der Frontkamera aufgezeichnet wurden. Es wurde versucht, typische Flugsituationen zu filmen. Diese wurden dann geschnitten und kategorisiert. Um diesen Prozess so kurz wie möglich zu halten, wurden die ersten Videos nicht mit dem Copter gefilmt, sondern die Kamera per Hand geführt. Die Resultate dieser Methode waren nicht sehr zufriedenstellend, da es äußerst schwer ist, die Kamera beim Laufen ruhig zu halten und keine rhythmischen Wackler bei jedem Schritt im Video zu haben. So entschied ich mich doch für die Variante, die Daten mit dem Copter zu sammeln. Dafür mussten Umbauten an ihm vorgenommen werden, um Kollisionen fliegen zu können ohne den Quadrocopter zu gefährden.
Der Einfachheit halber wurden die Kategorien zuerst auf drei Fälle beschränkt, um deren Unterscheidbarkeit zu testen. Diese gliederten sich in:

- Kollisionsflüge
- Links an Hindernissen vorbei
- Rechts an Hindernissen vorbei

Da sich eine Erhebung von Daten für ausreichend viele Situationen als zu aufwendig darstellte, wurde in einer späteren Phase der Arbeit ein leicht abgeänderter Ansatz gewählt. Für diesen werden nur Kollisionsflüge benötigt. Selbst für Kollisionsflüge gibt es verschiedene Arten von Situationen, für die Daten erhoben werden müssen. Dabei müssen verschiedene Arten von Objekten und Geschwindigkeiten beachtet werden.

5.3 Die Vorverarbeitung der Daten

Die Daten mussten daraufhin in die Kategorien eingeteilt und auf die essentiellen Stellen zugeschnitten werden. Der Ansatz dieser Arbeit beruht darauf, Muster im Optischen Fluss wiederzuerkennen und einzuordnen. Um dies machen zu können, ist es hilfreich, feste Positionen der Features zu verwenden, da andererseits die Muster, je nach Umgebung und Featurepositionen, anders aussehen würden. Eine Wiedererkennung wäre so nicht möglich. Ich löste dieses Problem durch die Verwendung des Durchschnitts der Flüsse im Zentrum des entsprechenden Bildausschnittes. So erhält man gleich positionierte Flussvektoren, da die Bildausschnitte für jedes Bild gleich definiert sind. Durch die Mittellungen der Flüsse werden zudem Ausreißer zum Großteil herausgefiltert.

So wird jedes Bild der Videos in 5x5 Teilbilder beziehungsweise Bildausschnitte unterteilt und zwischen jedem aufeinanderfolgenden Bildpaar der durchschnittliche Fluss der Teilbilder berechnet. Dafür werden in jedem Bildausschnitt gute Features gesucht. Dies geschieht mithilfe der OpenCV Implementierung der Shi-Tomasi Eckenerkennung(Kapitel 3.1), der Funktion *goodFeaturesToTrack*[odtb]. Hierfür ist es notwendig, das Bild vom RGB-Format zur Grauskalierung zu konvertieren, da der Algorithmus mit Farbbildern nicht umgehen kann. Es werden in jedem Bildausschnitt 40 prägnante Punkte gesucht, was insgesamt 1000 Punkte ergibt. Die Parametrisierung des Algorithmus wurde hier experimentell bestimmt.
Sie lauteten wie folgt:

- minDistance = 0.8
  Dieser Parameter gibt den minimalen euklidschen Abstand zwischen den einzelnen Features an.
- qualityLevel = 0.005
  Dieser Parameter bestimmt die Mindestqualität eines Features. Der Wert wird mit dem besten Qualitätswert multipliziert und ergibt so den Mindestwert für die Qualität eine Features, das akzeptiert wird.
- blockSize = 5
  Dieser Parameter gibt die Größe des Blocks um ein Feature an, der verwendet wird, um die in Kapitel 3.1 beschriebene Gleichung zu lösen. Eine Blockgröße von 5 bedeutet, dass ein 5x5 Pixel großes Feld um jedes Feature verwendet wird.

Um den Optischen Fluss zu erhalten, werden die wahrscheinlichsten Positionen der Punkte in den entsprechenden Bildausschnitten des Folgebildes berechnet. Dies geschieht mithilfe der ebenfalls in OpenCV implementierten

Funktion *calcOpticalFlowPyrLK* [odta]. Diese verwendet die Lucas-Kanade-Methode(Kapitel 3.2), um den Fluss zu berechnen. Nun wird aus den entsprechenden Flüssen der Durchschnitt für jeden Bildausschnitt gebildet. Diese Flüsse stellen nun die Ground Truth-Daten dar. Diese habe eine Dimensionalität von 50, da es 25 Bildausschnitte gibt mit jeweils einem x- und y-Wert.

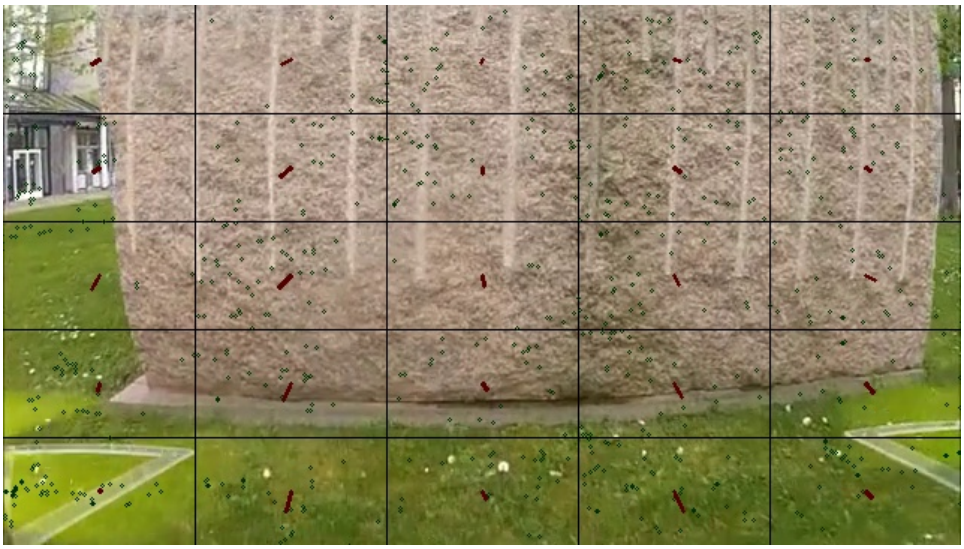

Abbildung 5: Bild eines Kollisionsflugs mit Durchschnittsflüssen der Teilbilder.

Da der Optische Fluss oft Fehler beinhalten kann, z.B. durch ungünstige Lichtverhältnisse oder falsch wiedererkannte Punkte, werden die Daten nun geglättet. Dies geschieht durch Konvolution und einen zugehörigen Kernel:

$$[\frac{1}{8}, \frac{1}{8}, \underbrace{\frac{1}{2}}_{\text{Anteil des zu glättenden Punkts}}, \frac{1}{8}, \frac{1}{8}]$$

Um die Effizienz der Verarbeitung der Daten während des Fluges möglichst hoch zu halten und die Qualität der Einschätzung zu verbessern, wird die Dimensionalität reduziert. Die Dimensionen sind im Fall dieser Arbeit die Flussvektoren der Bildausschnitte. Ein Datenpunkt hat zweimal so viele Dimensionen wie es Bildausschnitte gibt, da jeder Flussvektor aus einer x- und y-Komponente besteht, welche als einzelne Dimensionen des Punktes betrachtet werden.
Um eine Steigerung der Effizienz und der Generalisierung zu erreichen, wird die Hauptkomponentenanalyse, kurz PCA, verwendet (siehe Kapitel 3.3). PCA ermöglicht es, die Dimensionalität auf 33 Hauptkomponenten zu reduzieren. Die Ground Truth-Daten werden nun auf die durch die PCA erhaltenen Hauptachsen umgeformt und auf die ersten 33 Hauptkomponenten

projiziert. Die Anzahl der Hauptkomponenten, die verwendet werden, wurden Anhand einer Statistik ermittelt, die das Verhältnis von der Anzahl der Hauptkomponenten gegenüber dem Score der richtig erkannten Bilder darstellt. Es wird dabei angestrebt, möglichst wenig Hauptkomponenten zu verwenden, ohne eine starke Minderung der Erkennungsrate zu erhalten. Ohne die Reduzierung der Dimensionen könnte eine spätere Kategorisierung anhand der Daten zu speziell und eingeschränkt sein.
Nun wird aus den Daten für jede Kategorie eine multivariate Normalverteilung erstellt. Im Flug sollen anhand dieser die Wahrscheinlichkeitsdichten der aktuellen Daten berechnet werden. Aufgrund der Dichten wird dann entschieden, was für eine Situation vorliegt. Dafür wird experimentell ein Schwellwert festgelegt, wie groß der Unterschied zwischen den Dichten sein muss, und eine Mindestdichte, ab der eine Aussage signifikant genug ist. Genaueres dazu in Kapitel 5.4. Damit ist die Vorverarbeitung der Ground Truth-Daten abgeschlossen.

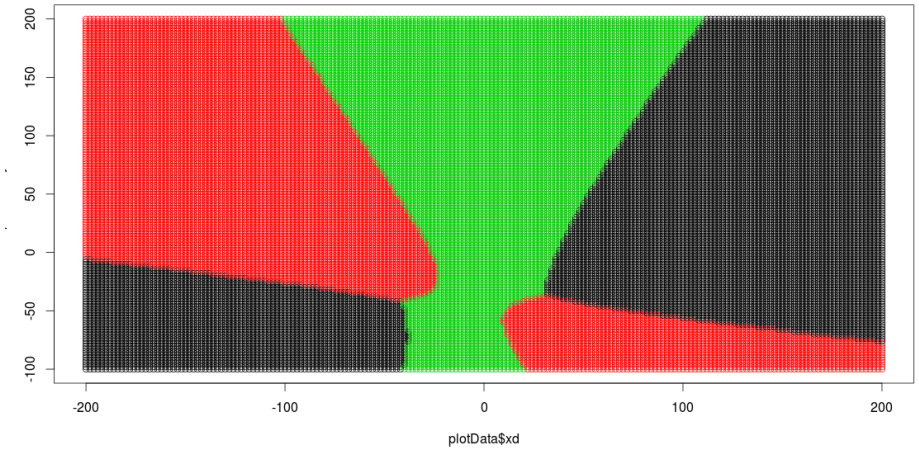

Abbildung 6: Kategorisiert man jeden Punkt mithilfe der Normalverteilungen, erhält in den ersten beiden Dimensionen folgende Einteilung. Für diesen Plot wurden die ersten aufgenommen Daten verwendet. Die Grüne Fläche sind als Frontalflug eingeordnete Punkte. Rot steht für links vorbeiführende Flüge und Schwarz für die rechts vorbeiführende Flüge.

5.4 Die Einschätzung von Situationen

Im Flug gilt es nun die Daten zu verwenden, um die Situationen einzuschätzen. Eine simple Lösung wäre eine Kategorisierung mithilfe des euklidischen Abstands zum Zentrum der Ground Truth-Daten. Hierbei ignoriert man allerdings wichtige Eigenschaften der Punktwolke, zum Beispiel die

unterschiedlich starken Ausdehnungen in verschiedene Richtungen. Zudem können Ausreißer das Ergebnis stark verzerren. Um diesem Problem entgegenzuwirken, wird in dieser Arbeit angenommen, dass die Daten normalverteilt sind und man somit eine Normalverteilung zum Einschätzen verwenden kann. Dafür werden alle wichtigen Daten im Code der Kollisionserkennung definiert. Dazu gehören die Werte für die Umformung auf die Hauptachsen, also die Rotationsmatrix und eine Skalierungsmatrix. Weiterhin werden die Parameter der multivariaten Normalverteilungen benötigt, welche diese definieren. Diese wurden zuvor aus den Ground Truth-Daten berechnet.

Während des Fluges werden dann der Optische Fluss für jedes Bild berechnet. Dies geschieht mit dem gleichen Verfahren wie für die Ground Truth-Daten, sodass man wieder die Durchschnittsflüsse der Bildausschnitte erhält. Auch diese werden geglättet, wie in der Vorverarbeitung der Ground Truth-Daten, allerdings mit einem anderen Kernel, der nur rückblickend arbeitet. Dieser lautet wie folgt:

$$[\frac{1}{4}, \frac{1}{4}, \underbrace{\frac{1}{2}}_{\text{Anteil des zu glättenden Punkts}}]$$

So erhält man für das aktuelle Bild den Datenpunkt, der nun kategorisiert werden kann. Dafür muss der Punkt noch auf die Hauptachsen umgeformt werden. So umgeformt kann man mithilfe der Normalverteilungen die entsprechenden Dichten für ihn berechnen und mithilfe der vorher festgelegten Schwellwerte entscheiden, ob es sich um eine Kollision oder um eine ungefährliche Situation handelt. Dabei gibt es zwei mögliche Ansätze: Entweder man modelliert ausreichend viele Situationen aus vorher dafür aufgenommenen Ground Truth-Daten oder man verwendet nur Kollisionsdaten und ermittelt einen Schwellwert, ab dem eine Kollision vorliegt. Der erste Ansatz ist sehr viel aufwendiger als der zweite. Die umfangreichere Modellierung bietet allerdings die Möglichkeit, auch andere Situationen zu erkennen, die vielleicht später für andere Aufgaben relevant werden könnten.

6 Evaluierung

Um den Ansatz zu evaluieren, testete ich ihn in mehreren Schritten; zunächst mit wenigen Daten und auf eine spezifische Situation begrenzt, dann ausweitend auf die realen Gegebenheiten. Genauer gesagt gab es zwei sehr ähnliche Ansätze. Der erste versuchte, alle möglichen Standardsituationen zu modellieren und zu unterscheiden. Der zweite reduzierte die Situationen auf mögliche Kollisionen und Nicht-Kollisionen.

6.1 Daten und Zuverlässigkeit

Die ersten Tests mit Daten, die sehr ähnlich zu den Ground Truth-Daten waren, erzielten gute Ergebnisse. Dafür wurden Daten für drei Kategorien verwendet und nur ca. 350 Datenpunkte. Diese sind aufgeteilt in Kollisionsflüge, links vorbeiführende und rechts vorbeiführende Flüge. Einen Datenpunkt bilden hierbei alle vorverarbeiteten durchschnittlichen Flüsse der Bildausschnitte eines Bildes. Im Plot der ersten beiden Hauptkomponenten (siehe Abbildung 7) kann man die Trennung der Daten in bestimmte Gebiete gut erkennen.
In den Experimenten mit diesen Daten wurden die Bilder noch in 10×10 Teilbilder unterteilt. Später reduzierte ich die Anzahl auf 5x5 Teilbilder. So können bessere Durchschnittsflüsse berechnet werden, da mehr Features per Teilbild gefunden werden können, weil diese eine größere Fläche pro Ausschnitt besitzen. Die gute Erkennungsrate sieht man deutlich im Plot8 des Verhältnisses von richtig erkannten Daten zu verwendeten Hauptkomponenten. Mit 50 Hauptkomponenten erreicht man so eine 80 prozentige Erkennungsrate.
Für den Einsatz in einer reellen Umgebung waren diese Ground Truth-Daten nicht ausreichend, da die Daten zu wenige der möglichen Situationen abdeckten. So erstellte ich mehr Ground Truth-Daten, um eine bessere Modellierung der realen Verhältnisse zu erreichen. Ein großes Problem stellen verschiedene Geschwindigkeiten dar, da sich nicht nur die Längen der Flüsse verändern, sondern, da der Copter einen größeren Winkel bei höheren Geschwindigkeiten annimmt, auch das Muster bei Kollisionen. Versucht man möglichst viele Geschwindigkeiten beziehungsweise Winkel bei der Datenaufnahme zu beachten, erhält man schwerer voneinander zu trennende Kategorien, wie man im Plot 9 der ersten und dritten Hauptkomponente gut erkennen kann.
Um dem Problem der sich stark überlappenden Kategorien entgegenzuwirken, wurde der Ansatz abgeändert. Es werden nun Kollisionen nur mithilfe eines Schwellwertes und den Daten für Kollisionen bestimmt. So wird ignoriert, dass es noch andere Fälle als *Kollisionen* und *Nicht-Kollisionen* gibt. Hierfür wurden insgesamt 2455 Datenpunkte für Kollisionen erstellt. Das Problem der unscharfen Definition der Kollisionsdaten ließ sich so allerdings

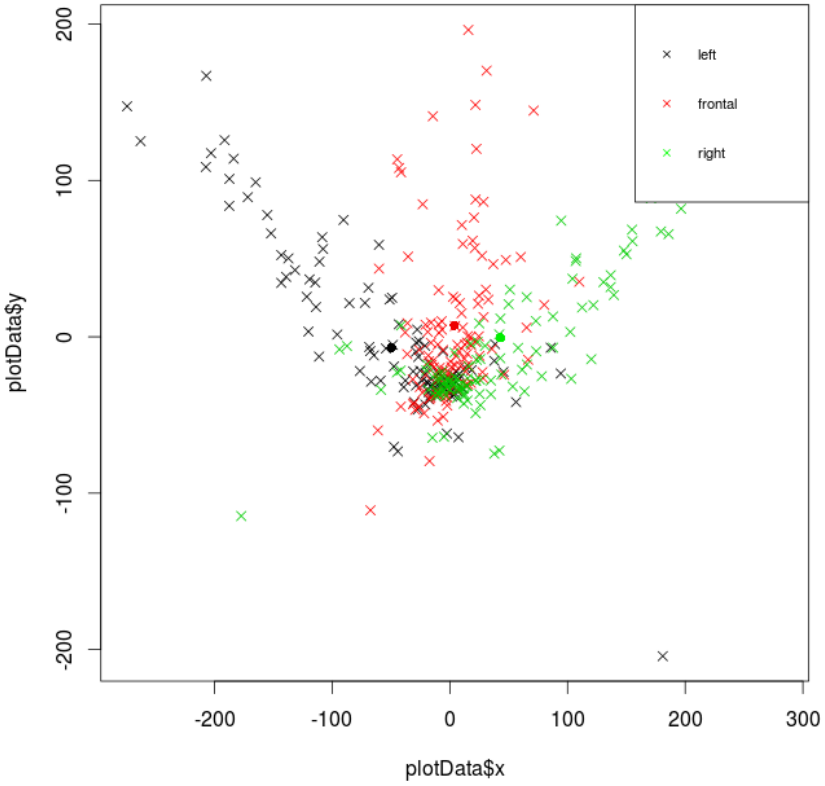

Abbildung 7: Plot der ersten beiden Hauptkomponenten der ersten Datenerhebung. Die Daten enthalten noch keine unterschiedlichen Winkel, sondern sind per Hand aufgenommene Flüge, wie man sie von einem Flugzeug erwarten würde. Die Kreise zeigen die Mittelpunkte der Punktwolken.

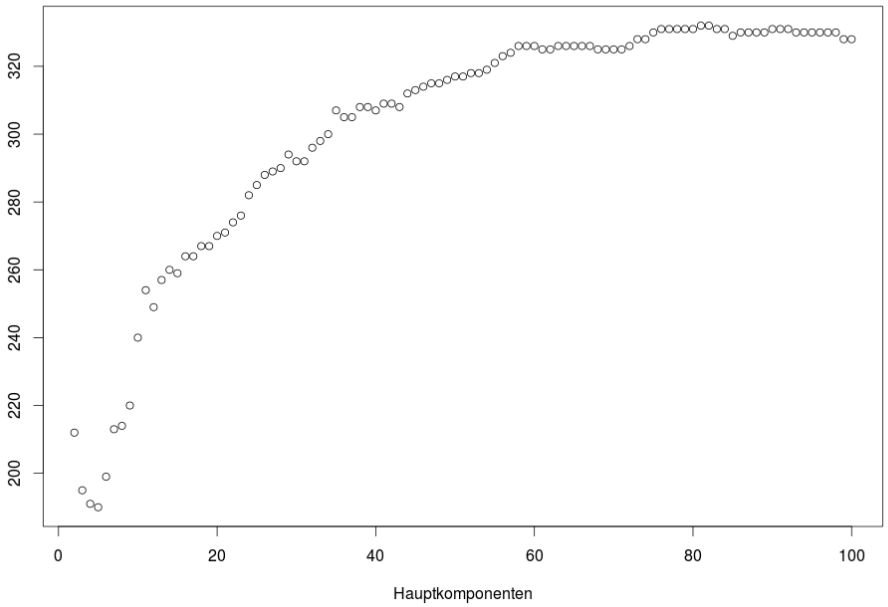

Abbildung 8: Plot von Hauptkomponenten zu richtig erkannten Bildern. Es wurde gegen ein Set von 340 Bildern getestet.

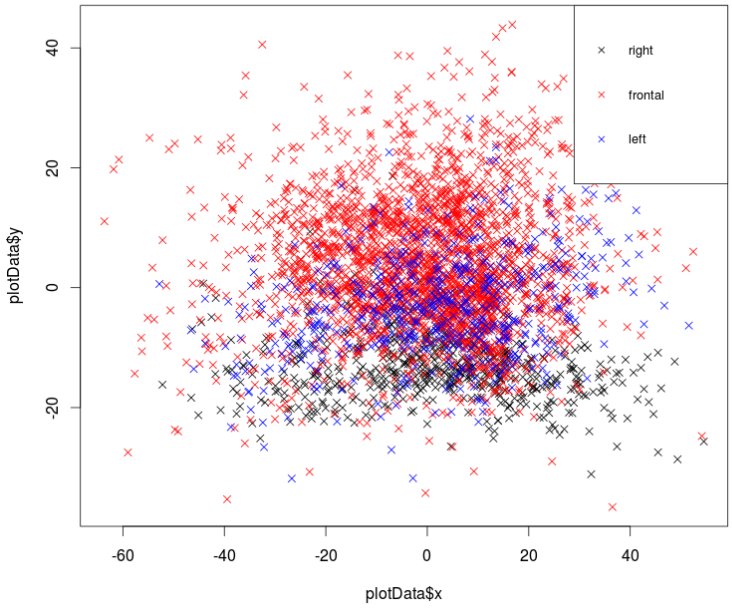

Abbildung 9: Plot der ersten und dritten Hauptkomponente mit weiter modellierten Kollisionsdaten. Man sieht deutlich, wie sich nun die Daten überlagern, da weitaus mehr verschiedene Situationen für die Kollisionen modelliert wurden.

nicht beheben. Ein Plot von richtig erkannten Daten zu verwendeten Hauptkomponenten10 zeigt die teilweise schlechten Erkennungsraten.

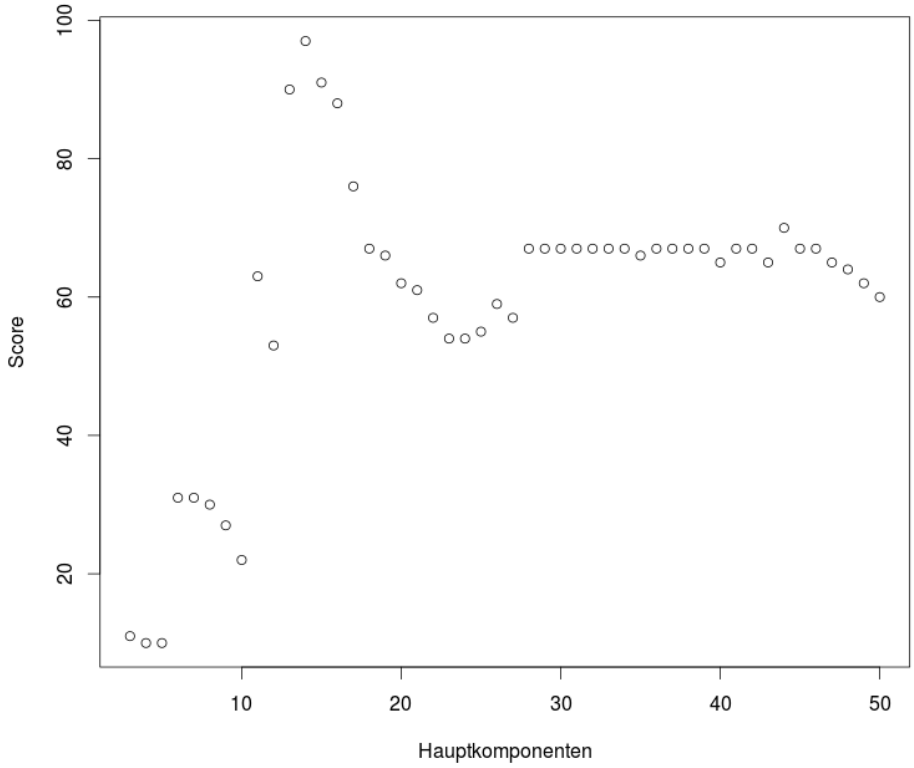

Abbildung 10: Plot von Hauptkomponenten zu richtig erkannten Kollisionen für Ground Truth-Daten mit vielen verschiedenen Kollisionsvarianten und Geschwindigkeiten, diesmal gegen ein Testmenge der Größe 140.

Um falsch erkannte Kollisionen zu reduzieren und die Effizienz zu erhöhen, wird nur eine Kategorisierung vorgenommen, wenn die Geschwindigkeit des Neurocopters einen Schwellwert übersteigt. Die Geschwindigkeit wird in dieser Arbeit durch die Gesamtlänge des Flusses bestimmt. Weitere genauere Berechnungen sind möglich durch das Gyroskop oder den Optischen Fluss einer auf den Boden gerichteten Kamera.
Weiterhin wird in dieser Arbeit von einer Normalverteilung der Daten ausgegangen. Da diese Annahme nicht vollständig der Realität entspricht, kann hier mit mehr Nachforschungen eine besser zutreffende Verteilung gefunden werden. Damit würde sich dann auch die Erkennungsrate verbessern.

Die letzten Tests mit zirka 200 Testdaten ergaben eine Erkennungsrate von

41,4 Prozent bei 17 Hauptkomponenten. Die Testdaten wurden in natürlicher Umgebung mit dem Neurocopter erhoben. Ein Vergleich zu einer Klassifizierung mit dem euklidischem Abstand erreichte lediglich eine Erkennungsrate von 0,01 Prozent.
Ein zweiter Test mit anderen Daten bestehend aus 840 Bildern erreichte eine Erkennungsrate von 62,6 Prozent. Versucht man diese Daten durch den euklidischen Abstand zu kategorisieren, erhält man eine Erkennungsrate von 0,001 Prozent.

Die Testdaten enthalten noch Translations- sowie Rotationsbewegungen. Eine Eliminierung der Rotationsbewegungen war zur Zeit dieser Arbeit nicht mehr möglich. Allerdings bestätigten Experimente, bei denen die Kamera von Hand geführt wurde und die Daten zur Laufzeit ausgewertet wurden, dass eine Erkennung bei wenigen Rotationsbewegungen recht erfolgreich ist.

6.2 Effizienz

Eine richtige Effizienzeinschätzung ist schwierig, da es viele Parameter gibt, die die Effizienz beeinflussen, wie zum Beispiel die Anzahl der Features pro Teilbild, die Anzahl der Teilbilder, die Anzahl der Hauptkomponenten und einige mehr. Da für die Bildverarbeitung im Moment die Algorithmen nicht explizit parallelisiert werden, ist hier noch einiges an Verbesserung möglich. Das Odroid besitzt vier Kerne und ist somit prädestiniert für parallele Berechnungen.
Die letzten Messungen auf dem Odroid ergaben eine Berechnungsgeschwindigkeit von 24-29 Bilder pro Sekunde, was für den Einsatz im realen Umfeld bereits ausreichend wäre, solange die Erkennungsrate genau genug ist. Eine höhere Bilderrate ist anstrebsam, da so zum einen mehr Daten pro Sekunde verarbeitet werden können, womit eine Reduzierung für Fehleinschätzungen möglich wäre. Zum anderen ermöglicht eine höhere Verarbeitung eine schnellere Reaktion auf gefährliche Situationen.

7 Diskussion

Nach Abschluss der Arbeit stellten sich die Ergebnisse anders dar als erwartet. Es gab rückblickend mehrere schwerwiegende Probleme, die in der Planung nicht richtig abgeschätzt werden konnten. Zum einen war der Zeitaufwand für die Erhebung der Ground Truth-Daten sehr viel größer als vermutet. Die Herausforderung bestand darin, authentische Daten aufzunehmen, ohne viel fliegen zu müssen, da Kollisionsflüge eine nicht zu vernachlässigende Gefahr für Mensch und Maschine darstellen. Das Erstellen der Daten per Hand ist jedoch ebenso schwierig, da ein häufiges *Verwackeln* nicht zu verhindern ist. So waren mehrere Ansätze für eine gute Flugaufnahme nötig. Eine mögliche Lösung dieses Problems wäre der Bau einer Vorrichtung, wie

zum Beispiel eine Schiene mit Schlitten für die Kamera. Solch eine Vorrichtung sollte die Möglichkeit bieten, die Kamera in verschiedenen Winkeln zu positionieren und gleichmäßig zu bewegen. Die Herstellung solch einer Vorrichtung war allerdings im Rahmen dieser Arbeit nicht möglich.

Ein weiteres Problem stellen die stark variierenden möglichen Kollisionsflüge dar. Durch deren Vielfältigkeit wurde es zunehmend schwerer mit mehr aufgenommenen Situationen, eine gute Trennung zwischen Kollision und Nicht-Kollision zu erreichen. Hier hätte vielleicht eine abgeänderte Methodik geholfen. Es wäre zum Beispiel denkbar, eine andere Methode als die Hauptkomponentenanalyse zu verwenden, wie die *Exploratory Factor Analysis*. Durch die stärkere Neigung der Kamera bei zunehmender Geschwindigkeit des Copters ist in extremen Fällen ein rechtzeitiges Erkennen von Kollisionen nicht möglich, da die Hindernisse viel zu spät in den Sichtbereich der Kamera kommen, um noch reagieren zu können. So ist dieser Ansatz mit fest montierter Kamera nicht für hohe Geschwindigkeiten geeignet. Eine freischwingende Kamera beziehungsweise eine sich selbst in der Waage haltende Halterung könnte hier eine Lösung bieten.

Experimente zeigten, dass das System mit reinen Translationsbewegungen recht gute Erkennungsraten erreicht. Führte der Copter allerdings Drehungen aus, stieg die Anzahl an falsch erkannten Kollisionen stark an. Solche Rotationsbewegungen könnten durch Verwendung des Gyroskops erkannt werden. Diese können dann aus dem Fluss heraus berechnet werden. Ebenfalls wäre es möglich Situationen mit großen Winkelgeschwindigkeiten zu ignorieren. Im Endeffekt werden im Moment der Großteil der Kollisionen erkannt, allerdings auch recht viele falsch erkannte Kollisionen. Es gibt diverse Möglichkeiten, die Rate der *falsch Positiven* zu verringern. Diese werden im Kapitel 8 näher beschrieben.

Abschließend kann man sagen, dass der Ansatz für das Ziel dieser Arbeit genügt, allerdings fehlen noch einige Verbesserungen, um es in der Realität einsetzen zu können.

8 Ausblick

Es gibt noch viele Dinge, die an diesem Ansatz verbessert werden können, aber im Rahmen dieser Arbeit nicht mehr umgesetzt werden konnten. So zum Beispiel die in Kapitel 7 angesproche Vorrichtung zum Erstellen von Ground Truth-Daten. Ebenfalls wäre ein Algorithmus, der stark unsaubere Daten manuell aussortiert, eine starke Erleichterung der Datenerhebung. Eine weitere mögliche Verbesserung wäre die Einführung eines Zweikamerasystems, wie von Moore at al. in ihrem Paper [MTB+09] beschrieben. Dies

würde auch eine Eliminierung von Rotationen im Optischen Fluss ermöglichen, welche im momentanen System zu Fehlern führen. Das Filtern und Glätten der Daten kann auch noch weiter ausgebaut werden, um falsche Erkennungen zu vermeiden. Da bisher nur Konvolution und Mittelung zum Glätten verwendet wurden, kann hier noch viel verbessert werden. Eine Möglichkeit wäre an dieser Stelle zum Beispiel der RANSAC Algorithmus.
Da die verwendeten Algorithmen für die Berechnung des Optischen Flusses auf der Basis von grauskalierten Bildern arbeiten, könnte hier der Wechsel zu Algorithmen, die Farbbilder unterstützen, zu Verbesserungen führen.

Zusammenfassend kann man sagen, dass der in dieser Arbeit vorgestellte Algorithmus mit weiteren Verbesserungen eine passable Lösung für das Problem der Kollisionserkennung darstellt.

9 Glossar

- Time-To-Contact: Die Zeit bis zur Kollision mit einem Objekt bei gleichbleibender Geschwindigkeit und gleichbleibendem Kurs.
- Focus-Of-Expansion: Das Zentrum des Optischen Flusses. Der Punkt, von dem die meisten Flussvektoren weg zeigen.
- Feature: Ein prägnanter Punkt, der zur Berechnung des Optischen Flusses verwendet wird.
- Drohne: Ein autonom fliegender Roboter.
- Multicopter: Ein Helikopter mit einer Mehrzahl von Rotoren.
- Round Robin: Methode des Scheduling. Prozesse bekommen nacheinander zirkulär Rechenzeit beziehungsweise Ressourcen.
- OpenCV: Open-Source Framework, für die Verarbeitung von Bildern und Videos. Bietet eine umfangreiche Bibliothek an mathematischen Werkzeugen.
- Ground Truth Daten: Daten die zum Lernen von Systemen des Überwachten Lernens verwendet werden. Sie bilden die „Wahrheit“ des Systems.

Literatur

[co.] Hardkernel co. Odroid-U3. http://www.hardkernel.com/main/main.php. [Online; accessed 23-July-2014].

[GO08] William E Green and Paul Y Oh. Optic-flow-based collision avoidance. *Robotics & Automation Magazine, IEEE*, 15(1):96–103, 2008.

[Hei] UniversitÃ¤t Heidelberg. Karlheinz Meier. http://www.kip.uni-heidelberg.de/user/meierk/research/projects. [Online; accessed 23-July-2014].

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

[HS88] C Harris and M Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[LK81] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, May 1981.

[MTB+09] Richard JD Moore, Saul Thurrowgood, Daniel Bland, Dean Soccol, and Mandyam V Srinivasan. A stereo vision system for uav guidance. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3386–3391. IEEE, 2009.

[NBR+13] Emre Neftci, Jonathan Binas, Ueli Rutishauser, Elisabetta Chicca, Giacomo Indiveri, and Rodney J Douglas. Synthesizing cognition in neuromorphic electronic systems. *Proceedings of the National Academy of Sciences*, 110(37):E3468–E3476, 2013.

[odta] opencv dev team. Lucas-Kanade OpenCV Docs. http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html. [Online; accessed 31-July-2014].

[odtb] opencv dev team. Shi-Tomasi OpenCV Docs. http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html. [Online; accessed 31-July-2014].

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[Pro] Human Brain Project. Human Brain Project. https://www.humanbrainproject.eu/. [Online; accessed 23-July-2014].

[SB12] Kenneth Sebesta and John Baillieul. Animal-inspired agile flight using optical flow sensing. *CoRR*, abs/1203.2816, 2012.

[Sig] Sigbert. PcaIdea. http://commons.wikimedia.org/wiki/File:PcaIdea.png#mediaviewer/Datei:PcaIdea.png. [Online; accessed 14:28, 30 October 2009 (UTC) - Eigenes Werk. Lizenziert unter Public domain Ã¼ber Wikimedia Commons].

[SK07] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4(1):13–16, 2007.

[Sri11] Mandyam V Srinivasan. Honeybees as a model for the study of visually guided flight, navigation, and biologically inspired robotics. *Physiological Reviews*, 91(2):413–460, 2011.

[ST93] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Ithaca, NY, USA, 1993.

[sto] UAV store. ArduPilot Mega. http://www.uav-store.de/autopilot-1/apm/. [Online; accessed 23-July-2014].

[UF] Berlin United-FUmanoids. Berlin United-Framework. http://http://www.fumanoids.de/code/framework/. [Online; accessed 21-August-2014 (UTC)].