**Julius-Maximilians-Universität Würzburg**
Fakultät für Mathematik und Informatik

**Informationstechnik für Luft- und Raumfahrt**
Lehrstuhl für Informatik 8 Prof. Dr. Sergio Montenegro


# Bachelorarbeit

## Punktgenaues Starten und Landen eines Quadrocopters mittels optischer Sensorik

Vorgelegt von

Matthias Büttner

Matr.-Nr.: 1746505

Prüfer: Prof. Dr. Sergio Montenegro

Betreuende wissenschaftliche Mitarbeiter: Dipl.-Ing. Nils Gageik

Würzburg, 13. 09. 2013

# Erklärung

Ich versichere, dass ich die vorliegende Arbeit einschließlich aller beigefügter Materialien selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.
Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Werken entnommen sind, sind in jedem Einzelfall unter Angabe der Quelle deutlich als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form noch nicht als Prüfungsarbeit eingereicht worden.
Mir ist bekannt, dass Zuwiderhandlungen gegen diese Erklärung und bewusste Täuschungen die Benotung der Arbeit mit der Note 5.0 zur Folge haben kann.

Würzburg, 13. 09. 2013

Matthias Büttner

# Aufgabenstellung

Die Fortschritte im Bereich Sensorik und Mikrotechnik ermöglichen heutzutage den kostengünstigen Bau kleiner unbemannter Luftfahrzeuge (UAV, unmanned airial vehicle, Drohne) wie Quadrokopter sowie deren Ausstattung mit einer Vielzahl an Sensorik. Kameras sind heute für wenige Euros zu kaufen und stellen eine sehr mächtige Sensorik dar. Während es Systeme für die ferngesteuerte Anwendung schon zu kaufen gibt, ist die Forschung im Bereich autonomer Systeme noch in den Anfängen. Insbesondere der Indoor-Betrieb ist aufgrund fehlender absoluter Positionsstützung durch GPS problematisch. Der Aufbau eines eigenen autonomen Systems wird daher am Lehrstuhl Aerospace Information Technology der Uni Würzburg erforscht und erprobt. Im Rahmen dieses Forschungsvorhabens ist das bestehende System weiterzuentwickeln, so dass die Genauigkeit bei Start und Landung erhöht wird.

Das aktuelle System regelt seine Position mittels Kamera und Bestimmung des optischen Flusses. Es kann seine Höhe regeln und autonom starten und landen. Rotationen und Höhenänderungen führen zu Fehlern bei der Positionsbestimmung. Dies ist insbesondere beim Starten ein Problem. Die Auswirkungen dieser Fehler sind durch ein geeignetes Verfahren zu reduzieren.

Das aktuelle System kann darüber hinaus mit Hilfe einer auf den Boden gerichteten WebCam Objekte, d.h. Bälle, erkennen und deren Mittelpunkt und Radius bestimmen. Nun sollen diese Informationen in einem zu entwickelnden Verfahren verwendet werden, damit der Quadrokopter genau über einem solchen gefundenen Objekt autonom landen kann.

Eine ausführliche Übersicht über den Stand der Technik und vergleichbare Ansätze ist dabei zu erstellen. Die entwickelte Software ist in das bestehende System einzubinden und soll an diesem ausgiebig evaluiert werden. Zur Aufgabe gehört eine ausführliche Dokumentation.

Aufgabenstellung (Stichpunktartig):

- Stand der Technik: Punktgenaues Landen mit optischer Sensorik
- Landung über rotem Punkt (Ball) implementieren
- Kompensation von Höhen- und Orientierungsänderungen im Falle OF-Positionsbestimmung und Positionsbestimmung des roten Punktes (Balles)
- Evaluierung
- Dokumentation

# Zusammenfassung

Das Thema dieser Arbeit ist die Implementierung einer punktgenauen Landung in Verbindung mit der Optimierung des genutzten Optischen-Fluss-Sensors. Mittels einer bereits vorhandenen Objekterkennung soll ein Ball gefunden werden, dessen Mittelpunkt als Landeziel dient. Um dies umzusetzen, muss dieser Mittelpunkt aus der Klasse der Objekterkennung extrahiert werden und mit den aktuellen Positionsdaten des Quadrocopters zu einem Wegpunkt zusammengeführt werden. Dabei werden aktuelle Höhe und Neigung berücksichtigt und herausgerechnet. Ist der Wegpunkt gesetzt, wird dieser mit Hilfe des Optischen-Fluss-Sensors, welcher auch um eine Höhen- und Neigungskompensation erweitert wird, angeflogen. Über dem Bestimmungsort angekommen, wird die Landung eingeleitet, mit dem Ziel punktgenau auf dem Ball zu landen.

# Inhaltsverzeichnis

# 1 Einleitung

Kleine und wendige Drohnen sind in der heutigen Zeit nichts Ungewöhnliches. Zumeist werden sie für Kartografierungs- und Aufklärungsflüge genutzt oder aber auch um Rettungskräfte in Krisensituationen mit wichtigen Informationen zu versorgen. Solche UAV (Unmanned Aerial Vehicle) werden durch moderne Technik immer leistungsfähiger, was sie mittlerweile in die Lage versetzt während des Fluges komplexe Aufgaben, wie Objekterkennung, zu bewältigen. Dies kann in Zukunft genutzt werden, um in für den Menschen schwierig oder sogar unzugänglichen Gebieten nach zum Beispiel Überlebenden oder gefährlichen Stoffen zu suchen. Systeme, die vollkommen autonom agieren sind allerdings wenig erforscht und aus diesem Grund kaum verbreitet.

Der Lehrstuhl für Informatik 8 der Universität Würzburg hat sich zur Aufgabe gesetzt ein solches, komplett autonomes UAV, in Form eines Quadrocopters, zu entwickeln. Es ist bereits in der Lage beispielsweise eigenständig eine Suche nach Objekten durchzuführen, Hindernisse zu erkennen oder Wegpunkten zu folgen und im Anschluss zu landen. Die Methode, die hierbei zur Landung zum Einsatz kommt, erlaubt jedoch nur eine relativ geringe Genauigkeit von circa 30cm. Um dies zu verbessern wird in dieser Arbeit ein Landesystem implementiert, mit dem Ziel eine punktgenaue Landung durchführen zu können und genauere Positionsdaten zu erhalten.

Dazu wird zu Beginn ein Einblick in das Prinzip der autonomen Landung und in die Grundlagen der vom Lehrstuhl 8 genutzten Objekterkennung gegeben. Im Anschluss wird eine genaue Positionierung über einem Landeziel (Ball), Verbesserung der Positionserkennung via Optischem-Fluss-Sensor und die Landung an sich beschrieben, indem zunächst das Konzept vorgestellt, danach die Implementierung erklärt und am Ende evaluiert wird. Letztendlich werden die Ergebnisse diskutiert und ein Ausblick auf Verbesserungen und mögliche weitere Entwicklungen gegeben.

# 2 Stand der Technik und Grundlagen

## 2.1 Prinzip der autonomen Landung

Ziel des Systems ist es autonom in Gebäuden zu fliegen, diese zu erkunden und mit den gesammelten Daten zurückzukehren. Um die Informationen sicher auslesen zu können, ist es wichtig eine selbstständige Landung zielgenau durchzuführen. Zur Zeit bestehen mehrere Ansätze, um dieses Problem zu lösen. Die verbreitetste Methode ist es, ein vorher bekanntes Muster zu finden sich anhand diesem genau über der Landestelle zu positionieren und die Landung auszuführen. Eine weitere Methode, die allerdings nur außerhalb von Gebäuden eingesetzt werden kann, ist die Positionierung via GPS. Diese Art hat für das hier genutzte Indoor-System keine Verwendung.

### 2.1.1 Landemustererkennung

Die genutzten Muster sind sehr häufig einfache Schablonen mit einem schwarzen Muster auf weißem Hintergrund (siehe Abbildung 2.1), die an der Landeposition ausgelegt werden und auf denen gelandet wird. Diese Art der Positionierung wird oft genutzt, da sie meist sehr genau ist und das Muster leicht erkannt werden kann. Diese Variante arbeitet nach der Objekterkennung. Ein Problem hierbei ist

die Erkennung bei niedrigen Höhen, da dort Aufgrund des Öffnungswinkels der Kamera Teile des Musters nicht mehr zu sehen sein können.

Um dies zu umgehen ist die Nutzung eines sich wiederholenden Musters, wie zum Beispiel Kreise (siehe Abbildung 2.2)(vgl. Lange et al. [2008]). Durch die erkannten Muster kann anschließend ein Steuerbefehl gesendet werden, der den Quadrocopter mittig über dem Objekt positioniert. Bei der Landung wird kontinuierlich die Position korrigiert, um ein bestmögliches Ergebnis bei der Landung zu erzielen.

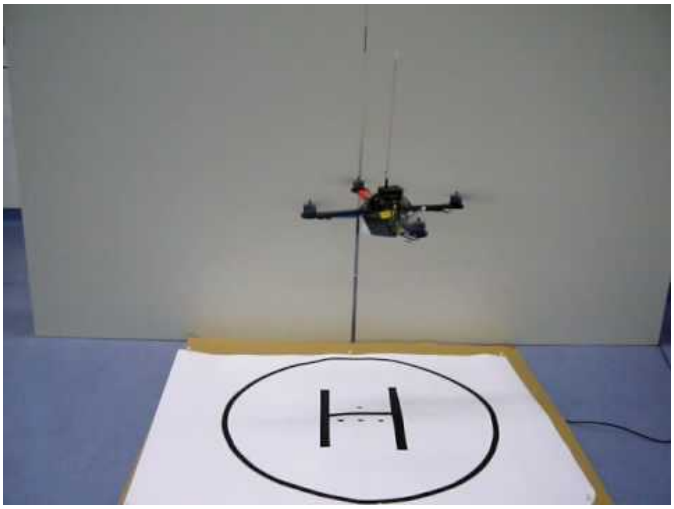


**Abbildung 2.1:** Landeschablone schwarz/weiß
Quelle: Wenzel et al. [2013]

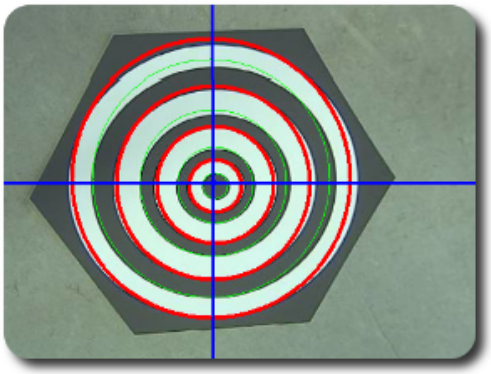


**Abbildung 2.2:** Kreismuster
Quelle: Lange et al. [2008]

Wie aus der Methode bereits zu erkennen, kommt hier eine Kamera zum Einsatz. Die kontinuierlichen Bildberechnung erfordert ein hohes Maß an Rechenleistung, weshalb diese auf einen externen PC ausgelagert wird. Die beiden Optionen hier sind, einen PC zur Berechnung direkt mitzuführen oder aber das System so auszulegen, diese auf einem externen Rechner durchzuführen.

Im folgenden wird auf den Landealgorithmus im Artikel "*Autonomous Landing for a Multirotor UAV*"Lange et al. [2008] eingegangen. Hier wird im speziellen Fall folgende Reihenfolge eingehalten (hier verkürzt dargestellt):

1. Bildaufnahme mittels Webcam
2. In binäres Bild umwandeln
3. Rauschen entfernen
4. Mittelpunkt und Kreisradien berechnen
5. Abstand in Metern zum Bildmittelpunkt errechnen
6. Quadrocopter positionieren

Bei der dort genutzten Methode besteht zudem die Möglichkeit anhand der erkannten Kreisradien die Höhe zu bestimmen. Die genutzte Webcam versendet ihre Bilder über WiFi und erlaubt somit eine Bildberechnung an einem PC, der nicht am Quadrocopter mitgeführt werden muss, bringt aber den Nachteil mit sich, dass das System nicht vollkommen autonom genutzt werden kann. Mittels dieser Berechnung erhält man die Höhe über und die relative Position zu dem Landepad, womit man problemlos die Position mittig anfliegen kann.

### 2.1.2 Infrarot-LED

Eine weitere Möglichkeit besteht darin, auf einen Infrarotsensor zurückzugreifen. Dieser Sensor soll nach einem bestimmten Schema angeordnete LED's erkennen

und danach die Positionierung vornehmen. Das Prinzip der Positionierung ist dabei das selbe wie bei 2.1.1.

Hierbei ist zu beachten, dass jegliche Infrarotstrahlung die Genauigkeit der Positionierung sehr stark beeinflussen kann. Sonnenstrahlen und Feuer sind die größten Fehlerquellen. Aus diesem Grund kann man diese Art der Positionierung nur in gut abgedunkelten Räumen nutzen, da sonst durch den Lichteinfall zu große Ungenauigkeiten entstehen können.

## 2.2 Objekterkennung

Der folgende Paragraph stützt sich auf die Arbeit *Implementierung und Evaluierung einer Objekter- kennung für einen Quadrocopter* von Christian Reul [2013]. Für die genaue Positionierung über dem Landeziel ist eine präzise Positionierung darüber erforderlich. Hierzu wird eine Objekterkennung bestehend aus einer Kamera und einem Windows-PC genutzt.

Um ein Objekt erkennen zu können, muss zuerst ein Scan durchgeführt werden. Dieser Scan dient als Referenzgegenstand und liefert zu suchende Farben, Radien und weitere Parameter. Anhand dieses Schemas wird nun nach Objekten mit ähnlichen Eigenschaften gesucht. Um diese zu finden, wird mit Hilfe einer Webcam ein Stream aus Bildern erzeugt und an den PC gesendet. Diese Bilder werden mittels Bildbearbeitung (Graustufenberechnung, Binärisierung) und geeigneter Algorithmen nach der Vorgabe durchsucht. Falls ein solches, ähnliches Objekt gefunden wird, berechnet das Programm den Mittelpunkt und speichert es in einen temporären Speicher.

# 3 Konzept

## 3.1 Landung mittels Ballerkennung

### 3.1.1 Überblick

Zur Landung mit Hilfe eines Balles, wird auf ein bereits bestehendes System zurückgegriffen. Dieses ermöglicht es via WebCam Bälle im Bildbereich der Kamera zu finden. Diese Arbeit beschäftigt sich damit, anhand der Mittelpunktsdaten der gefundenen Bälle und der aktuellen Höhe den Quadrocopter zentral über dem gefundenen Ball zu positionieren und die Landung einzuleiten. Hierbei bestehen verschiedene Möglichkeiten die Landung durchzuführen:

- Landen in Intervallen
- Landen in einem Zug
- Sofortiges Landen

Weiterhin gibt es diverse Ansätze die initiale, zentrale Positionierung des Quadrocopters vorzunehmen:

- Kontinuierliche Positionierung
- Einmalige Positionierung über Wegpunkt

Bei der autonomen Landung, um genauer zu sein der zentralen Positionierung über dem Landeobjekt, werden zudem anhand der Lagedaten des Quadrocopters (Quaternionen) die Roll- und Pitch-Winkel bestimmt. Dies dient der genaueren Erkennung des Balles und somit der genaueren Berechnung der Sollposition.

Außerdem fließt bei der Bestimmung der Positionsabweichung die aktuelle Höhe des Quadrocopters mit ein. Um eine Skalierung abhängig von der Höhe vorzunehmen, wird experimentell bestimmt, wie breit der erfasste Bereich, bei verschiedenen Höhen, am Boden ist. Anhand dieser Versuchsdaten kann ein Skalierungsfaktor bestimmt werden, aus dem die richtige absolute Abweichung mittels der WebCam-Daten berechnet werden kann.

### 3.1.2 Landen in Intervallen

Die Methode *Landen in Intervallen* folgt der Idee der stufenweisen Absenkung und periodischen Neupositionierung des Quadrocopters. Dabei wird wie folgt vorgegangen:

1. Zentrale Positionierung über dem Ball
2. Höhe verringern (zum Beispiel um 20cm pro Vorgang)
3. Schritte 1 und 2 wiederholen bis bestimmte Höhe über Boden unterschritten wurde
4. Landung vornehmen

Durch die Einhaltung dieser Schritte soll eine sehr genaue Landung erzielt werden. Allerdings ist zu erwarten, dass mit diesem Verfahren sehr viel Zeit beansprucht wird.

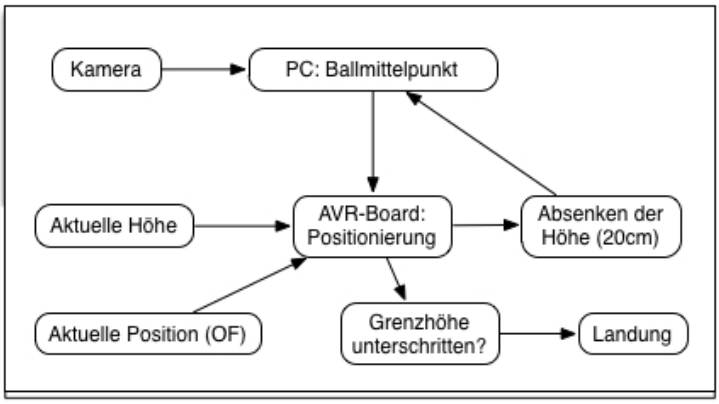



**Abbildung 3.1:** Blockdiagramm: Landen in Intervallen

### 3.1.3 Landen in einem Zug

Die Alternative *Landen in einem Zug* besitzt eine sehr ähnliche Vorgehensweise wie das *Landen in Intervallen*. Der große Unterschied ist, dass die Landung nicht stufenweise erfolgt, sondern die Schritte 2 und 3 komplett übersprungen werden. Hierdurch wird auf potentielle Genauigkeit beim Landen weniger Rücksicht genommen, aber die zur Landung benötigte Dauer soll erheblich verkürzt werden.

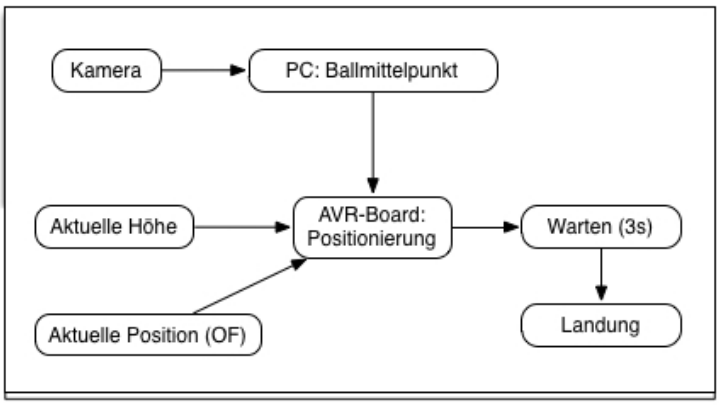



**Abbildung 3.2:** Blockdiagramm: Landen in einem Zug

### 3.1.4 Sofortiges Landen

Die letzte Möglichkeit *Sofortiges Landen* sieht vor, die Landung direkt nach dem Finden des Balles einzuleiten. Dies bedeutet, dass eine sehr ungenaue Landung zu erwarten ist, da der Quadrocopter sich möglicherweise noch in Bewegung befindet. Durch die Bewegung wirken auf den Quadrocopter noch horizontale Kräfte, wodurch er bei der Landung nicht gerade sinken wird, sondern sich von dem Fundort des Balles entfernt.

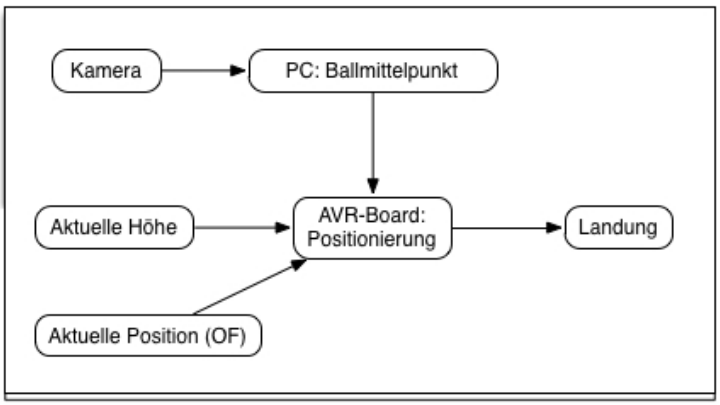



**Abbildung 3.3:** Blockdiagramm: Sofortiges Landen

### 3.1.5 Kontinuierliche Positionierung

Wie bereits beschrieben, existieren auch verschiedene Ansätze zur zentralen Positionierung über dem Ball. Die Variante *Kontinuierliche Positionierung* sieht dabei vor, dass der Ball während des Fluges kontinuierlich nachverfolgt wird. Aus den gewonnen Daten wird bei jeder Neuberechnung die aktuelle Position aktualisiert und soll somit eine direkte, mittige Positionierung ermöglichen. Da die Wegpunkte sehr genau angeflogen werden können, ist diese Methode höchstwahrscheinlich nicht von Nöten. Eine Positionierung wie in 3.1.6 sollte erfahrungsgemäß ausreichen, um eine hohe Genauigkeit zu erhalten.

### 3.1.6 Einmalige Positionierung über Wegpunkt

Bei der Taktik *Einmalige Positionierung über Wegpunkt* wird der Ball erkannt und die Sollposition des Quadrocopters auf die gefundene Stelle gesetzt. Durch die erwartete, hohe Genauigkeit der Kameradaten und der daraus resultierenden Präzession des Wegpunktes, wird nur einmal ein Wegpunkt gesetzt und das System fliegt diesen an.

In Verbindung mit der potentiellen Verbesserung der Daten aus dem Optischen-Fluss-Sensor, sollte diese Position sehr präzise angeflogen werden können. Eine weitere Korrektur der aktuellen Position oder des Wegpunktes ist in dieser Variante nicht vorgesehen.

## 3.2 Optischer Fluss Höhen- und Orientierungskompensation

Der Optische-Fluss-Sensor dient der genauen Positionsbestimmung des Quadrocopters. Um auf verlässliche Daten zurückgreifen zu können, muss dieser richtig Kalibriert sein. Um dies zu gewährleisten, werden dem Treiber dieses Sensors eine Höhenskalierung und Neigungskompensation eingefügt. Diese sind notwendig, da das System zum jetzigen Stand (August 2013) nur auf einer Höhe von einem Meter verlässliche Daten liefert und keine Winkeländerungen in die Positionsbestimmung mit einbezieht.

- Höhenskalierung:

  Bei der Höhenskalierung wird anhand der aktuellen Flughöhe ein Skalierungsfaktor eingefügt, der die gemessene optische Verschiebung in ein korrektes Verhältnis setzt. Um dies zu ermöglichen wird wieder experimentell in verschiedenen Höhen Messreihen erstellt. Anhand dieser Daten kann dann

ein Skalierungsfaktor ermittelt werden, der höhenabhängig die zurückgelegte Distanz in das richtige Verhältnis bringt.

- Neigungskompensation:

  Bei der Neigungskompensation werden auch einige Messreihen aufgenommen, die in Abhängigkeit der Roll- und Pitchwinkel die geänderte Position zeigen. Anhand der gewonnen Daten kann danach ein Faktor berechnet werden, der die Neigung in die verschiedenen Richtungen ausgleichen soll. Hierdurch soll die Position des Quadrocopters genauer bestimmt werden und Ungenauigkeiten minimiert.

# 4 Implementierung

## 4.1 Genutztes System

Zur Umsetzung des Konzeptes wird auf ein bestehendes System, den Quadrocopter, zurückgegriffen. Dieses ist bereits voll flugfähig und muss um die benötigten Klassen, Methoden und Nachrichten erweitert werden. Es wird versucht bereits bestehende Klassen so wenig wie möglich zu verändern. Falls nötig wird auf eine in dieser Arbeit erstellte Methode verwiesen, die dann ausgeführt wird und sich in einer separaten Klasse befindet. Hiermit wird gewährleistet, dass das vorhandene System nur minimal aus dem Ursprungszustand gebracht wird und alle Änderungen zur Fehlersuche leicht deaktiviert werden können.

Durch die Komplexität der auf dem Quadrocopter ausgeführten Software werden zwei verschiedene, unabhängige Programmträger verwendet:

- AVR-Board
- Windows-PC

Um jedoch die aufwändige Bildberechnung mit der aktuellen Position zu verbinden, wird eine Kommunikation benötigt, die durch eine USART-Verbindung zwischen den beiden Boards realisiert wird.

## 4.2 Software

### 4.2.1 Übersicht

Da, wie bereits genannt, für dieses System auf zwei verschiedenen Plattformen programmiert wird, wurde ein Teil des Codes, welcher auf dem AVR-Board läuft, in C geschrieben, der andere Teil, welcher auf dem PC des Quadrokopters läuft, in C++. Die Entwicklungsumgebungen waren AVR32-Studio (Version 2.6.0) für die C-Programmierung und QT Creator (Version 2.0.1) für die C++-Programmierung. Beide Programme werden bereits zur Verwaltung und Weiterentwicklung des Quadrokopters verwendet.

Im Folgenden werden die in Kapitel 3 vorgestellten Konzepte im Detail erläutert. Die in dieser Arbeit neu erstellten Klassen (Header- und Source), mit den benötigten Methoden, haben die Namen *Landing.h/c* (AVR32-Studio) bzw. *Landing.h/cpp* (QT Creator).

### 4.2.2 Landung mittels Ballerkennung - Mittelpunktserkennung

Die zur Mittelpunktserkennung benötigten Daten befinden sich bereits in der Klasse *objectRec.cpp*, aus der bereits erwähnten Bacheloararbeit Reul [2013] zur Objekterkennung. Um diese nun zu extrahieren und zur weiteren Berechnung nutzen zu können, wurden die Klassen *objectRec.cpp* und *Landing.cpp* mittels eines Signals verbunden. Diese Verbindung befindet sich in der *mainwindow.cpp*. Bei jeder Bildberechnung in *objectRec.cpp* wird, falls in dieser ein Kreisobjekt erkannt wurde, ein Signal emittiert, welches den Mittelpunkt des Kreises an *Landing.cpp* sendet, genauer gesagt an die Methode *Landing::calc(Circle)*.

Diese Daten können nun dort verwendet werden, um die Abweichung zum Bildmittelpunkt (in Pixel) nun wie folgt berechnen zu können:

$$\begin{aligned} x_{offset\_pxl} &= x_{center} - x_{kreis} - x_{kameraoffset} \\ y_{offset\_pxl} &= y_{center} - y_{kreis} - y_{kameraoffset} \end{aligned} \tag{4.1}$$

Somit erhalten wir den relativen Abstand zum Bildmittelpunkt in Pixel. Da das System in Metern rechnet, muss dieser Wert noch umgerechnet werden. Um dies zu ermöglichen wurde überprüft, ob sich die gesehene Breite linear zur Höhe verhält. Messungen in verschiedenen Höhen zeigen die gemessene Breite in Meter. Hierbei ergaben sich folgende Werte:

| **Höhe in m** | 1.00 | 0.80 | 0.60 | 0.40 |
|---|---|---|---|---|
| **Breite in m ± 0.01m** | 0.75 | 0.60 | 0.45 | 0.30 |

Anhand dieser Werte kann man sehr schön erkennen, dass ein linearer Verlauf existiert und somit ein einfacher Skalierungsfaktor eingeführt werden kann. Es wurde nun per Definition einer Höhe 1.0m die Breite von 0.75m zugewiesen, also ein Faktor von 0,75. Damit kann nun ein *Meter-pro-Pixel*-Faktor bestimmt werden, welcher sich nach der aktuellen Höhe in Meter richtet. Dieser Faktor ist bei einer Kameraauflösung von 192 Pixeln in der Breite:

$$meterPerPixel = currentHeight \cdot \frac{0.75}{192pxl} \approx currentHeight \cdot 0.0039\frac{1}{pxl} \tag{4.2}$$

Mittels diesem Faktor kann man nun sehr einfach die aus Formel 4.1 bestimmten Pixelwerte in Meter umrechnen:

$$\begin{aligned} x_{offset\_m} &= x_{offset\_pxl} \cdot mPerPixel \\ y_{offset\_m} &= y_{offset\_pxl} \cdot mPerPixel \end{aligned} \tag{4.3}$$

Die beiden Werte $x_{offset\_m}$ und $y_{offset\_m}$ geben nun die Abweichung vom Mittelpunkt der Kamera zum Ballmittelpunkt an.

### 4.2.3 Landung mittels Ballerkennung - Positionierung

Um nun die Positionierung des Quadrocopters vorzunehmen, sind einige Schritte notwendig:

1. Senden der aktuellen Position und Ausrichtung (Quaternion) vom AVR-Board an den PC
2. Berechnung der Abweichung wie in 4.2.2 beschrieben
3. Rückenden der berechneten Daten vom PC an das AVR-Board

#### 4.2.3.1 Lagedaten senden

Es besteht, wie bereits in 4.1 erläutert, eine generelle Kommunikationsmöglichkeit zwischen den beiden Boards über ein USART-Interface. Mittels eines definierten Message-Protocols, welches in der *protocol.h* implementiert ist, besteht die Option, Nachrichten mit einem festen Inhalt zu senden. Für die Zwecke dieser Arbeit wurde ein Packet mit dem Namen *QOPTER_DATA* angelegt, welches neben der aktuellen X- und Y-Position und der aktuellen Höhe auch die momentane Lage in Quaternionenform an den PC sendet.

Auf dem PC wurde die *Landing.cpp* so angelegt, dass sie Nachrichten vom Typ *QOPTER_DATA* empfängt und die enthaltenen Daten in lokale Variablen speichert. Nach dem Erhalt der verschiedenen Lagedaten werden auch aus dem Quaternion sofort die entsprechenden Eulerwinkel berechnet, die später zur Berechnung der korrekten Abweichung herangezogen werden.

#### 4.2.3.2 Berechnung und Rücksendung der Daten

Nach der Entgegennahme und ersten Verarbeitung der vom AVR-Board gesendeten Daten, werden diese mit den in 4.2.2 errechneten Werten zusammengeführt. Um ein gute Ergebnis zu erzielen, wird mittels der aktuellen Position und der Abweichung zum Ballmittelpunkt einmalig ein Wegpunkt berechnet und an das AVR-Board gesendet. Zu diesem Zweck wurde eine eigene Nachricht mit dem Namen *LANDING_POINT* angelegt, in welcher die Koordinaten des Wegpunktes übertragen werden.

Auf dem AVR-Board wird eben diese Nachricht in der *Messaging.c* entgegengenommen. Diese wurde erweitert, um den Wegpunkt richtig zu setzen und die autonome Landung einzuleiten. Das heißt wir weisen dem nächsten Wegpunkt die berechneten Koordinaten zu, aktivieren den automatischen Flug dorthin und halten dort die Position. Zugleich wird der Status gesetzt, der den Quadrocopter in die autonome Landung bringt.

Es besteht weiterhin die Möglichkeit nach dem Senden und Setzen des Wegpunktes die Position des Quadrocopters kontinuierlich in Bezug auf den Ball zu aktualisieren. Um diese Option zu nutzen wurde das *Define OF_POSITION_CORRECTION* eingefügt, welches diese an- und abschaltet.

Wenn sie genutzt wird, wird durch die weitere Berechnung der Abweichung des Bildmittelpunktes zum Ballmittelpunkt, der Offset berechnet und nicht mehr als *LANDING_POINT*, sondern als *QOPTER_POSITION_ERROR* gesendet. Diese Nachricht löst nun nicht mehr das Setzen eines Wegpunktes auf, sondern aktualisiert die Quadrocopterposition in Bezug auf den zuvor gesetzten Wegpunkt:

$$\begin{aligned} x\_pos &= x\_soll\_pos - x\_pos\_error \\ y\_pos &= y\_soll\_pos - y\_pos\_error \end{aligned} \tag{4.4}$$

Somit soll eine verbesserte Genauigkeit erreicht werden und den möglichen Fehlern des Optischen-Fluss-Sensors entgegenwirken.

### 4.2.4 Landung mittels Ballerkennung - Landung

Bei der Landung werden zwei verschiedene Konzepte implementiert:

1. Landung in einem Zug
2. Landung in Intervallen

Diese Varianten werden ja nach Auswahl ausgeführt, nachdem der Quadrocopter mittels der oben genannten Positionierung (4.2.2, 4.2.3) auf die richtigen Koordinaten navigiert wurde.

Zur einfacheren Auswahl der Landemethode wurde in der *Landing.h* des AVR-Codes ein *Define* eingefügt, welches die Intervalllandung aktiviert und zeitgleich die Landung in einem Zug deaktiviert. Ist das *Define* jedoch nicht gesetzt, wird die normale Standardlandung in einem Schritt gewählt. Nach jeder Änderung der Landemethode muss der Code neu auf das AVR-Board gespeichert werden, da keine Auswahl zur Programmlaufzeit unterstützt wird.

#### 4.2.4.1 Landung in einem Zug

Bei der Landung in einem Zug wird auf die bereits vorhandene Landeregelung zurückgegriffen. Im Folgenden wird dieser Algorithmus kurz erläutert.

In der Arbeit *Implementierung und Evaluierung einer Höhenregelung für einen Quadrocopter* "[...] wurde ein Landealgorithmus implementiert, der sich aus 3 Phasen zusammensetzt."(Rothe [2012]). Die drei Phasen gliedern sich wie folgt:

1. Sicherheits-Annäherung
2. Annäherung an die Lande-Phase

3. Endanflug

In Phase 1 fliegt der Quadrocopter auf eine Höhe von 60cm. Dies dient der Sicherheit und bremst den Quadrocopter in Landung etwas ab, bevor Phase 2 beginnt. In dieser wird eine neue Soll-Höhe von 10cm gesetzt und das System sinkt langsam weiter bis zu Phase 3. Hier wird sobald eine Höhe von 30cm (was sich im gerade noch messbaren Bereich befindet) das Fluggas kontinuierlich abgesenkt, bis die Landung vollzogen ist (vgl. Rothe [2012]).

In dieser Zeit wird keine weitere Positionskorrektur über dem Landeziel vorgenommen.

#### 4.2.4.2 Landung in Intervallen

Im Gegensatz zur in 4.2.4.1 vorgestellten Landemethode, wird nun eine weitere Variante implementiert, die eine intervallweise Landung vorsieht. Hierzu wurden drei weitere States in die Landeregelung eingefügt, die die ersten beiden Phasen der Landung in einem Zug ersetzen. Ein weiterer Unterschied ist, dass bei der Landung in Intervallen eine kontinuierliche Positionskorrektur erfolgt, da die Landung nur fortgesetzt wird, wenn sich der Quadrocopter im Toleranzbereich der Landung befindet.

Die drei neu eingefügten Status wiederholen sich bis eine Höhe von 30cm unterschritten wird. Sie werden, wie soeben erwähnt, auch nur ausgeführt beziehungsweise fortgesetzt, solange sich das System mittig über dem Ziel befindet. Sollte sich der Quadrocopter bei der Landung aus dem Toleranzbereich hinausbewegen, muss er zuerst seine Position korrigieren, bevor die Landung an der selben Stelle, an der sie unterbrochen wurde, wieder aufgenommen wird.

- *LANDING_SINK*

  In diesem Schritt, merken wir uns die aktuelle Höhe, da wir diese später zu Vergleichszwecken benötigen, in einer neuen Variable. Im Anschluss wird die

Sollhöhe des Systems auf die aktuelle Höhe abzüglich einer *Landing-Step-Weite*, welche auf 20cm pro Schritt definiert wurde, gesetzt. Sobald dies geschehen ist, springt das System in den nächsten Schritt.

- *LANDING_INITIATE_NEW_SEARCH*

  In diesem Schritt, wird dem PC mittels einer Nachricht mitgeteilt, dass erneut ein Wegpunkt über dem Ballmittelpunkt berechnet und gesetzt werden soll, um die kontinuierliche Positionskorrektur auf den aktuellen Stand zu bringen. Dieser Schritt dient ausschließlich dem Setzen eines neuen Wegpunktes und springt nun zum letzten der drei States.

- *LANDING_FINISH_SINK*

  Im letzten der drei Schritte, wird nun darauf gewartet, dass der Quadrocopter seine neue Sollhöhe erreicht. Sobald dies der Fall ist, wird die Schleife erneut durchlaufen, indem der Landestatus wieder auf *LANDING_SINK* zurückgesetzt wird.

Diese Prozedur wird so lange ausgeführt, bis der Quadrocopter eine Höhe von 30cm unterschreitet. Danach wird die Landung wie bei 4.2.4.1 zu Ende gebracht, indem direkt in die finale Landephase gesprungen wird, welche auch genutzt wird, sobald das System die Grenzhöhe erreicht hat.

## 4.3 Optischer Fluss - Fehlerkompensation

Der Optische-Fluss-Sensor ist einer der wichtigsten Sensoren zur Bestimmung der aktuellen Position. Da die Implementierung zur Zeit nur für eine Höhe von einem Meter kalibriert ist und keine Neigungsänderungen ausgleicht, kann der Quadrocopter auf anderen Höhen und bei starken Richtungsänderungen nur bedingt

operieren. Um diese Fehler zu minimieren, wird eine Höhen- und Neigungskompensation wie folgt eingefügt.

Alle hier aufgeführten Berechnungen befinden sich im AVR-Studio in der Datei *Landing.c*. Um die Optimierung der Optischen-Fluss-Daten vorzunehmen, wurde in *positionHold.c* kleinere Änderungen vorgenommen, die anstatt der direkten Integration der Sensor-Rohdaten, die hier implementierten Funktionen nutzt. Um beide Kompensierungen unabhängig voneinander testen zu können, wurden sie in separaten Methoden implementiert. Dies ermöglicht das einfache an- und abschalten mittels vordefinierter *Defines*.

### 4.3.1 Höhenkompensation

Wie das Konzept bereits vorsieht, wurde mittels einer Messreihe ein Faktor beziehungsweise Proportion bestimmt, der das Verhältnis zwischen Höhe und zurückgelegter Strecke in Optischen-Fluss-Einheiten angibt. Wie man Abbildung 4.1 und den Daten aus Tabelle 4.1 entnehmen kann, verhält sich die gemessene, zurückgelegte Strecke indirekt proportional zur Höhe.

| **Höhe in m** | 0.65 | 0.90 | 1,35 |
|---|---|---|---|
| **OF-Messung min** | -470 | -446 | -419 |
| **OF-Messung max** | -1302 | -1107 | -875 |
| **OF-Differenz (Betrag)** | 832 | 661 | 456 |

**Tabelle 4.1:** Markante OF-Daten aus Graph 4.1

Wenn man die Daten aus Tabelle 4.1 in einem Diagramm (siehe Abbildung 4.2) ausgibt, kann man die indirekte Proportionalität sehr gut erkennen. Da der Optische-Fluss-Sensor auf eine Höhe von einem Meter kalibriert ist, ist zur Kompensation des Höhenfehlers nur eine einfache Multiplikation mit der aktuellen Höhe

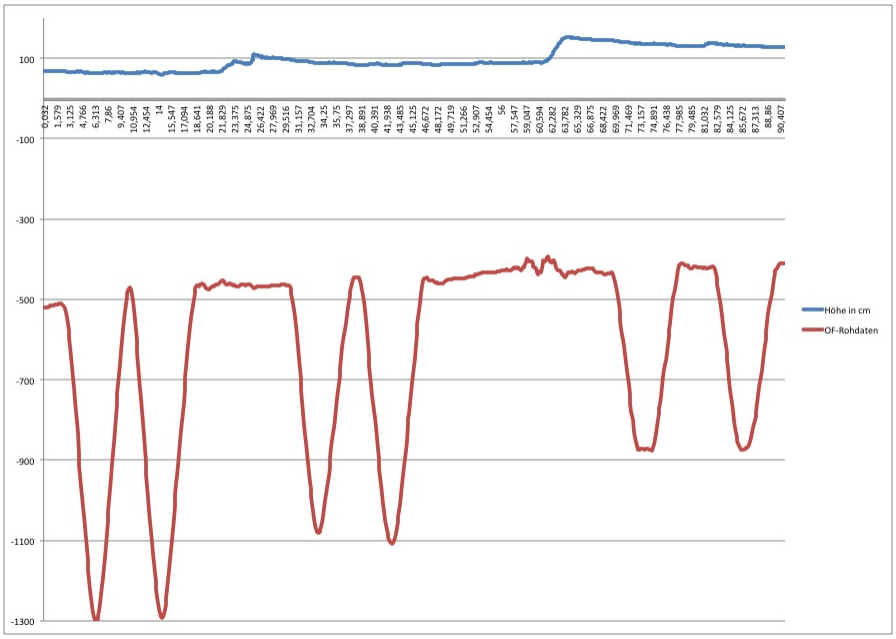



**Abbildung 4.1:** Optischer-Fluss-Sensor Rohdaten bei diversen Höhen

in Metern notwendig. Es ergeben sich somit folgende Berechnungen zur Kompensation der unterschiedlichen Höhen:

$$\begin{aligned} x\_pos &= x\_pos + sensor\_x_{raw} \cdot height \\ y\_pos &= y\_pos + sensor\_y_{raw} \cdot height \end{aligned} \tag{4.5}$$

### 4.3.2 Neigungskompensation

Bei der Implementierung der Neigungskompensation wurde Ähnlich zur Höhenkompensation vorgegangen. Als erster Schritt wurde die gedachte Positionsänderung bei verschiedenen Roll- und Pitchwinkeln gemessen. Anschließend wurde daraus ein Skalierungsfaktor bestimmt, der zusammen mit der aktuellen Höhe die Neigung in die entsprechende Richtung ausgleichen soll. Folgende Tabellen enthalten näherungsweise Angaben, da durch das Halten per Hand keine genauen Winkel gehalten werden konnten.

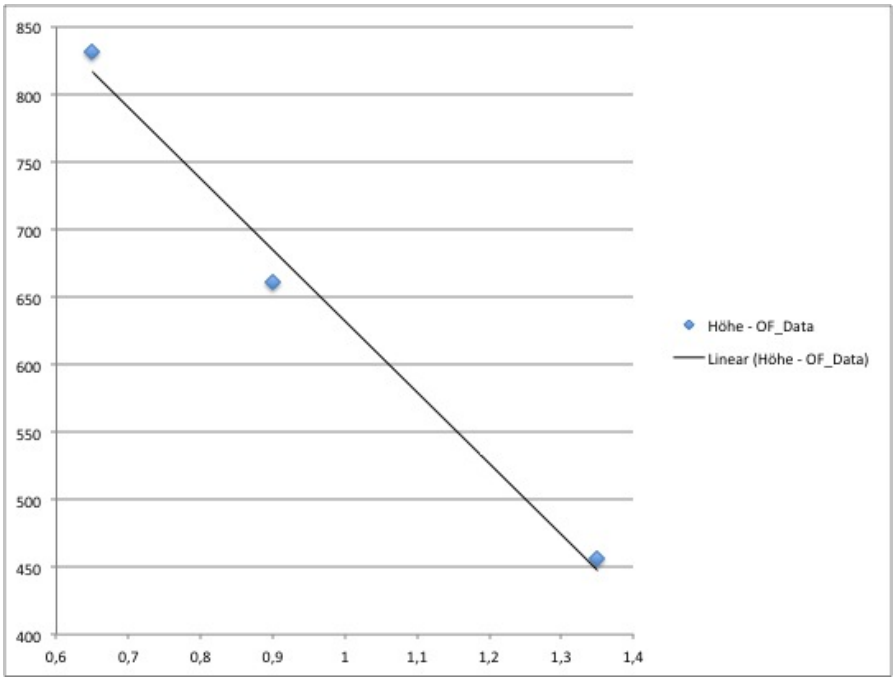



**Abbildung 4.2:** Optischer-Fluss-Sensor indirekte Proportionalität

| **Winkel in Grad** | -10 | -5 | 0 | 5 | 10 |
|---|---|---|---|---|---|
| **OF-Messung in cm** | 50 | 25 | 0 | -25 | -50 |

**Tabelle 4.2:** OF-Daten bei Pitch-Winkeländerung

| **Winkel in Grad** | -10 | -5 | 0 | 5 | 10 |
|---|---|---|---|---|---|
| **OF-Messung in cm** | -50 | -25 | 0 | 25 | 50 |

**Tabelle 4.3:** OF-Daten bei Roll-Winkeländerung

Anhand dieser Werte kann nun ein Kompensierungsfaktor *CM_PER_DEGREE* von $5\frac{cm}{Grad}$ abgelesen werden. Da wir durch den Flug starke Vibrationen erwarten, wurde ein Filter eingefügt, der kleine Winkeländerungen filtert, um möglichst nur die effektiven Änderungen der Winkel zu betrachten. Mittels des Kompensierungsfaktors und der aktuellen Höhe, kann nun einer unbeabsichtigten Positionsverschiebung durch Winkeländerung entgegengewirkt werden. Zum besseren

Debuggen, wurden noch zwei weitere Variablen zu den Sensorwerten hinzugefügt: *of_pos_x_roll_kompensation* und *of_pos_y_pitch_kompensation*.

$$\begin{aligned} of\_pos\_x\_roll\_komp &= delta\_roll \cdot CM\_PER\_DEGREE \cdot (höhe/100)/100 \\ of\_pos\_y\_pitch\_komp &= -delta\_pitch \cdot CM\_PER\_DEGREE \cdot (höhe/100)/100 \end{aligned} \tag{4.6}$$

Um nun diese Werte in die Position mit einfließen zu lassen, werden sie in jedem Durchlauf auf die aktuelle Position aufaddiert.

$$\begin{aligned} of\_pos\_x &= of\_pos\_x + of\_pos\_x\_roll\_komp \\ of\_pos\_y &= of\_pos\_y + of\_pos\_y\_pitch\_komp \end{aligned} \tag{4.7}$$

Mittels dieser beiden Methoden erhalten wir am Ende eine möglichst fehlerfreie Position aus dem Optischen-Fluss-Sensor.

# 5 Evaluierung

## 5.1 Landung mittels Ballerkennung

Zum genauen Positionieren müssen die Koordinaten des Ballmittelpunktes auch in Abhängigkeit der Höhe und Winkel bestimmt werden. Durch die eingefügte Höhenkompensation wird die Verschiebung des Balles, wenn der Quadrocopter nicht geneigt ist, richtig erkannt und Qualitativ sowie Quantitativ richtig ermittelt. Die aktuelle Höhe wurde auch in den weiteren Berechnungen immer auf aktiv gesetzt.

Beim Ausgleich der Pitch- und Roll-Winkel sind einige Ungenauigkeiten aufgetreten, da die Sensoren zur Bestimmung der Winkel zum Teil ungenaue Daten gesendet haben. Es führte oftmals zu genaueren Ergebnissen die Winkelkompensation zu deaktivieren, da sie die Position zu stark beeinflusst hat und . Aus diesem Grund und da beim Flug in der Phase, in der der Ball gefunden wurde, nur sehr kleine Winkel auftraten wurde diese Korrektur nicht weiter verwendet.

### 5.1.1 Positionierung über Landeziel

Die Positionierung über dem Landeziel hat, ohne Positionskorrektur durch die Kamera, sehr gut funktioniert. Wie aus Abbildung 5.1 zu sehen, wird nach der Ballerkennung die Sollposition auf die Mittelpunktskoordinaten des Balles gesetzt. Sobald dies geschehen ist, regelt das System die Position auf diesen Punkt. Ein

leichtes Überschwingen des Quadrocopters kann dennoch nicht verhindert werden, da durch das manuelle Ansteuern des Balles (bis dieser gefunden wurde) noch Steuerbefehle gesendet werden. Nachdem die keine weiteren Befehle durch die Remote-Verbindung gesendet werden, regelt sich der Quadrocopter sehr schön und innerhalb weniger Sekunden über den Punkt.

Der Positionsdrift im hinteren Teil des Graphen kommt von fehlerhaften Winkeldaten, die das System glauben lassen es hätte eine Neigung, die es ausgleichen müsse. Es wurde versucht, diesen Driftfehler mittels einer neuen, gedämpften Konstruktion des Gyro-Sensors zu minimieren. Deutlich ist jedoch zu erkennen, dass das setzen des neuen Zielpunktes korrekt ausgeführt wird und der Quadrocopter zu dieser Position geregelt wird.


**Abbildung 5.1:** Positionierung über dem Zielpunkt. Alle Werte in Metern.

Probleme bei der Positionierung traten vor allen Dingen in der richtigen Zuordnung der Positionsvariablen auf. Durch Versuche wurde die Größenordnung der

Daten gemessen und im Anschluss angepasst, um im jeweiligen Programm mit einer einheitlichen Größe rechnen zu können. Diese waren Meter im QT-Programm auf dem PC und Optische-Fluss-Einheiten auf dem AVR-Board. Weiterhin wurde zum Setzen des Wegpunktes experimentell die Vorzeichen der Abweichung zum Ballmittelpunkt ermittelt.

### 5.1.2 Positionskorrektur

Die implementierte Positionskorrektur mittels der Aktualisierung durch den Ballmittelpunkt hat zu großen Fehlern bei der Position geführt. Auf folgendem Graphen (Abbildung 5.2) sieht man, dass die Position nach dem Erkennen des Balles korrigiert wird, was den Quadrocopter dazu veranlasst seine Position zu Ändern und er außerhalb des Sichtbereiches des Balles kommt. Somit kann keine weiter Korrektur vorgenommen werden und das System behält den Offset bei. Was sich auch daran zeigt, dass der Quadrocopter nicht von sich aus die Landung einleitet und sich immer weiter von der Sollposition entfernt.

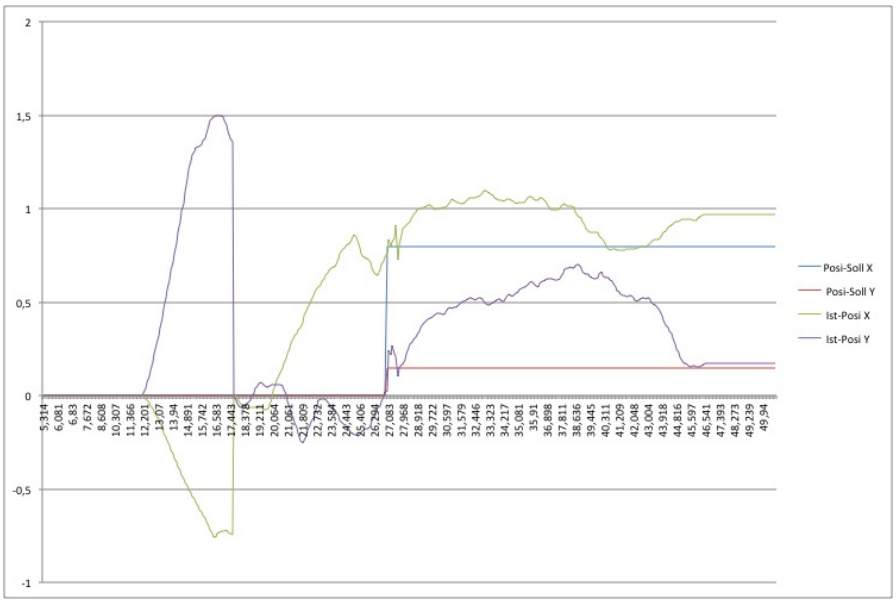



**Abbildung 5.2:** Fehler bei der Positionskorrektur durch Kamera

Dieser Fehler stammt mit großer Sicherheit aus der Verzögerung, die beim Senden, Berechnen und Rücksenden der Daten entsteht. Ein Ansatz, dies zu beheben, wäre einen Timestamp einzuführen, der Position nachträglich an der richtigen Stelle aktualisiert und den Delay somit entfernt.

### 5.1.3 Landung

Für die Landung an sich wurden nach der Positionierung drei verschieden Varianten vorbereitet. Aus technischen Gründen konnten nur zwei der Möglichkeiten getestet werden. Die beiden Optionen, welche getestet wurden, waren die direkte Landung und die Landung nach einem Zeitintervall. Die letzte der Möglichkeiten ist die intervallweise Landung, welche bereits im Code vorgesehen und implementiert ist. Es ist festzustellen, dass die Landung nach einem Zeitintervall sehr gute Resultate erzielt hat.

#### 5.1.3.1 Direkte Landung nach Finden des Balles

Bei dieser Art der Landung wurde sofort nach der Entdeckung des Balles die bereits implementierte Landesequenz eingeleitet. Da der Quadrocopter zu diesem Zeitpunkt allerdings noch in Bewegung war, ist er bei der Landung, wie erwartet, von der Zielpositon abgedriftet, was keinerlei Verbesserung zur Genauigkeit des bisherigen Landevorgangs darstellt, allerdings befindet sich die Landezone nun um das identifizierte Objekt. Um eine punktgenaue Landung zu erreichen, ist diese Art nicht geeignet.

In Abbildung 5.3 sieht man nun die Abweichung von der Sollposition nach der Landung. Diese kommt, wie bereits beschrieben, durch die restliche Bewegung des Systems vor dem Finden des Balles zustande. Um diesen Drift zu minimieren, wurde eine Wartezeit von drei Sekunden in die Landeregelung gesetzt (siehe 5.1.3.2).

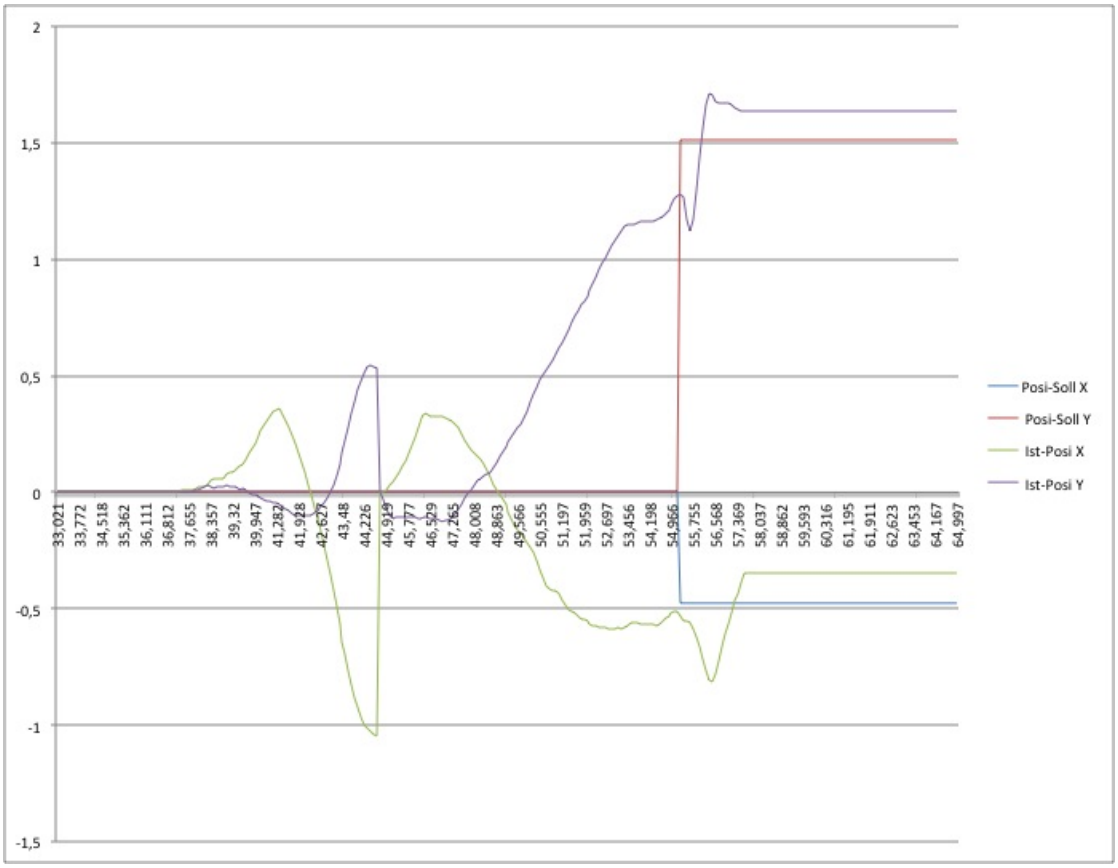



**Abbildung 5.3:** Landung direkt nach Erkennung des Balles. Alle Werte in Metern.

#### 5.1.3.2 Landung nach einem Zeitintervall

Nun wurde zur Landung ein Timer implementiert, der eine Landung erst ermöglicht, sobald drei Sekunden (per *Define* gesetzt) verstrichen sind und sich das System gleichzeitig im Toleranzbereich der Landung befindet. Durch das Einfügen des Timers soll gewährleistet werden, dass der Quadrocopter seine Bewegung weitestgehend gestoppt hat und die zentrale Position über dem Ziel eingenommen hat. Mit Hilfe des Timers ist es gelungen, den Flug des Quadrocopters vor dem Landen weitestgehend zu stoppen, was in einer wesentlich genaueren Landung resultierte.

Aufgrund von Ungenauigkeiten und Fehlern im Sensor, ist es jedoch nur in circa 50% der Fälle gelungen, eine genaue Landung zu erreichen. Bei den erfolgreichen

Versuchen lag die Landegenauigkeit in einem Radius von ungefähr sechs bis acht Zentimetern. Dies stellt eine deutliche Verbesserung zu den Landungen ohne diese Implementierung dar.

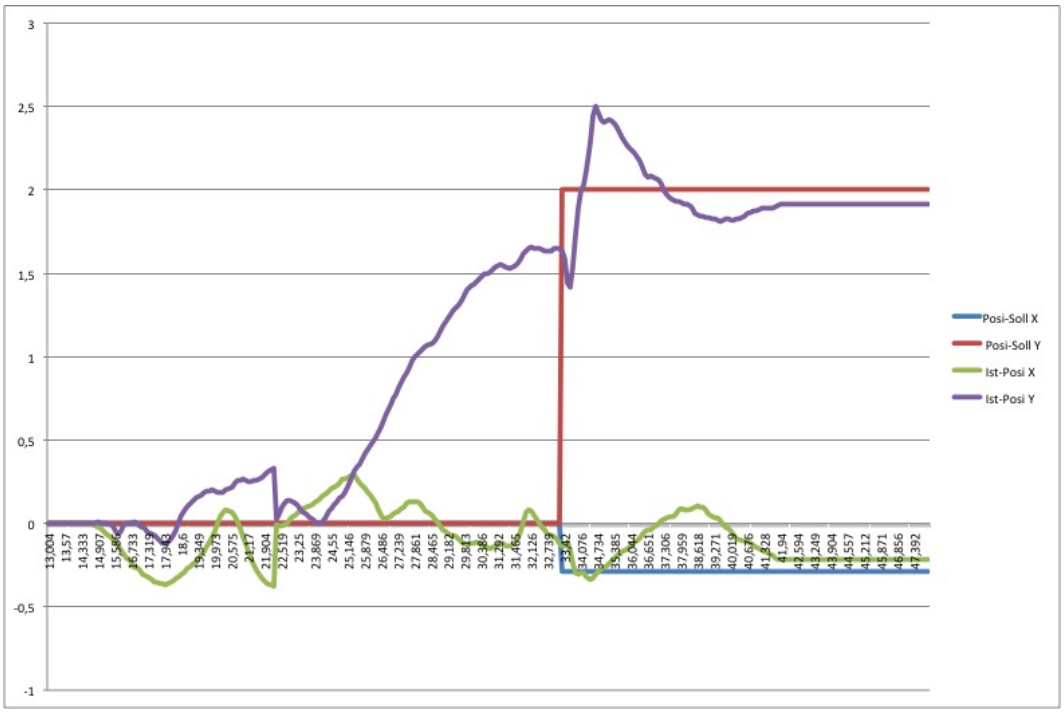



**Abbildung 5.4:** Landung 3s nach Erkennung des Balles. Alle Werte in Metern.

Durch die Landung wurden die X- und Y-Positionen des Systems etwas verfälscht. Der Quadrocopter ist allerdings bis auf wenige Zentimeter genau auf dem Ziel gelandet (siehe Tabelle 5.1). Um diese Abweichung weiter zu verbessern, wurde noch der Offset der Kamera in die Methode eingesetzt. Diese belaufen sich auf circa 2 Zentimeter in X-Richtung und circa 7,5 Zentimeter in Y-Richtung. Diese Werte decken sich auch sehr gut mit den Differenzen der Abweichungen aus Tabelle 5.1.

Potential zur Verbesserung zeigt sich hier im Bereich der exakteren Positionierung in Verbindung mit einer nahezu bewegungsfreien Positionshaltung vor der Landung.

| | zum Kameramittelpunkt | zum Quadrocoptermittelpunkt |
|---|---|---|
| Abweichung X in cm | 8 | 6 |
| Abweichung Y in cm | 3,5 | 15 |

**Tabelle 5.1:** Punktgenaue Landung - Messung der Abweichung per Hand

#### 5.1.3.3 Landung in Intervallen

Aufgrund technischer Schwierigkeiten mit den Sensoren des Quadrocopters und möglicher Fehler bei der Landung konnten keine Tests zu dieser Landemöglichkeit durchgeführt werden. Die Evaluierung des Codes legt nahe, dass die Landung durchgeführt werden kann. Das Einfügen eines so kritischen Bereiches des Fluges, bereitet aus Sicht der Sicherheit einige Komplikationen.

Es musste insbesondere darauf geachtet werden, die manuelle Landemöglichkeit durch diesen Zustand nicht zu blockieren. Die Methode zur Landung in Intervallen wurde aus diesem Grund noch einmal überarbeitet, um Abstürze bei der Landung definitiv ausschließen zu können.

## 5.2 Optischer Fluss - Fehlerkompensation

Die Fehlerkompensation beim Optischen-Fluss-Sensor hat eine große Verbesserung der Genauigkeit mit sich gebracht. Vor allem zeigt sich ein deutlicher Unterschied mit ein- und ausgeschalteter Höhenkorrektur.

### 5.2.1 Höhenkompensation

Die einfache Implementierung der Höhenkorrektur bei der Positionsbestimmung hat eine große Wirkung gezeigt. Um den Unterschied zu verdeutlichen, wurden mehrere Messungen durchgeführt:

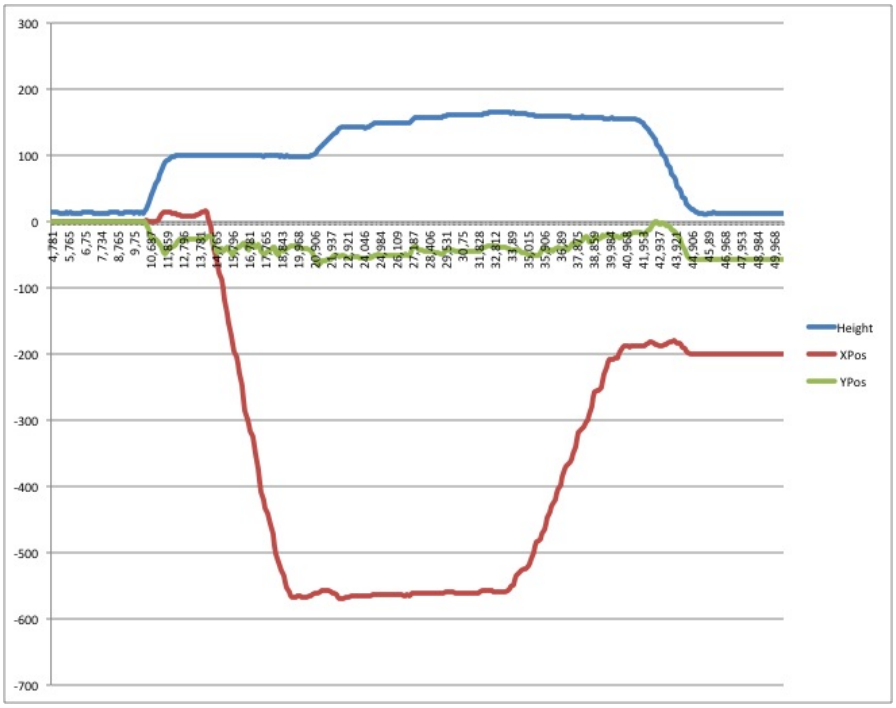



**Abbildung 5.5:** Hin-/Rückweg bei verschiedenen Höhen per Hand, Höhenkorrektur aus

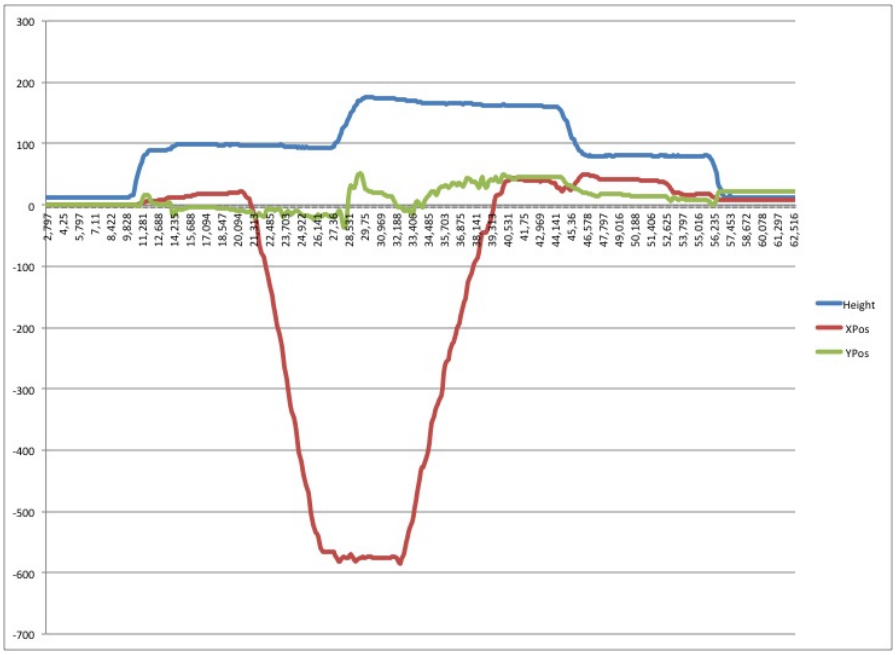



**Abbildung 5.6:** Hin-/Rückweg bei verschiedenen Höhen per Hand, Höhenkorrektur an

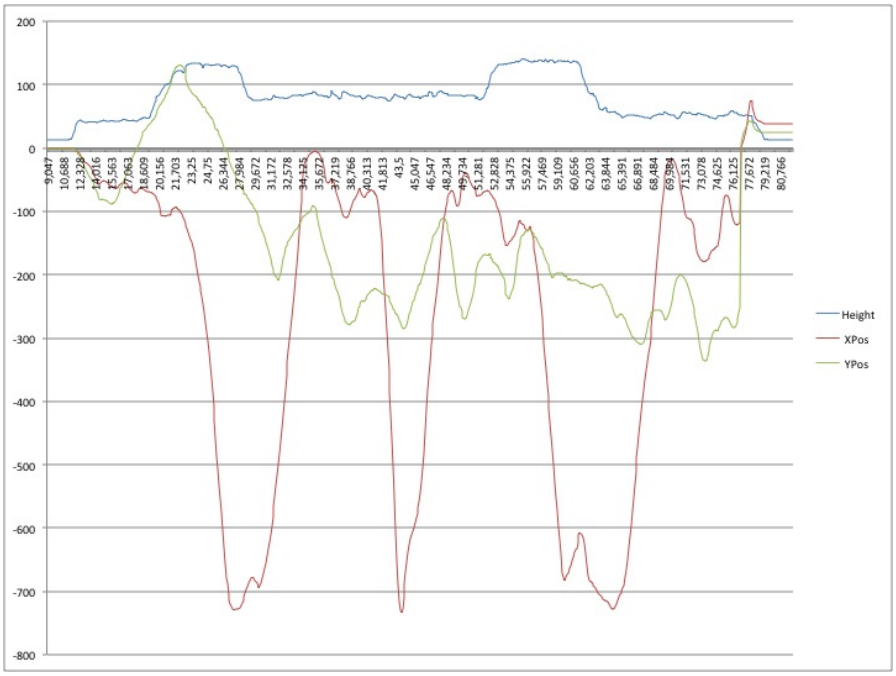



**Abbildung 5.7:** Hin-/Rückweg bei verschiedenen Höhen im Flug, Höhenkorrektur an

Ein Vergleich zwischen Abbildung 5.5 und Abbildung 5.6 zeigt sehr deutlich die Differenz zwischen der ein- und ausgeschalteten Kompensation. Es wurde jeweils versucht die selben Versuchsbedingungen zu schaffen, um einen besseren Vergleich ziehen zu können.

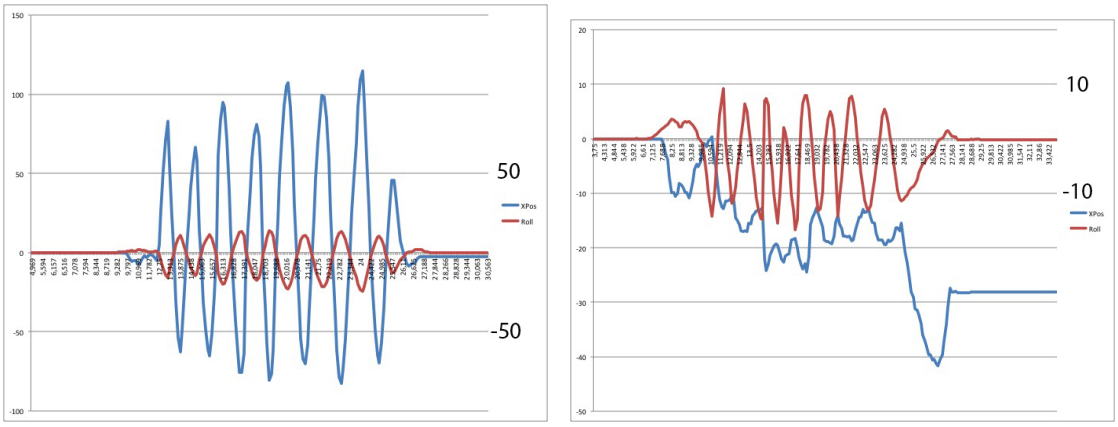
Der Versuchsaufbau war den Quadrocopter auf einer Höhe von circa einem Meter eine Strecke x zu bewegen, dort angekommen die Höhe auf ca. eineinhalb Meter zu erhöhen und die selbe Strecke wieder zurückzubewegen. Der Graph aus Abbildung 5.5 zeigen, dass das System nach der Rückkehr zum Ausgangspunkt von einer deutlich anderen Position ausgeht, als sie vorher der Fall war. Diese Abweichung kommt dadurch zustande, dass die Optischen-Fluss-Daten immer auf einen Meter skaliert werden, egal auf welcher Höhe sich der Quadrocopter befindet. Im zweiten Plot aus Abbildung 5.6 sieht man, dass mit aktiver Höhenkorrektur die Ausgangsposition wieder erreicht wird, obwohl die Hin- und Rückbewegung in verschiedenen Höhen stattgefunden hat. Die Abweichung hier ist entstanden, da auf

dem Rückweg eine etwas zu große Strecke gegangen wurde und somit auch eine andere Position angezeigt wird.

Abbildung 5.7 zeigt die Höhenkompensation aktiv im Flug. Auch hier wurde versucht den Quadrocopter die selbe Strecke auf einer Höhe hin- und auf einer anderen zurückfliegen zu lassen. Wie man sehr schön erkennen kann, kommt die Position jedes mal bis auf die ursprüngliche zurück, obwohl es durch die Flugdynamik schwer war immer die selbe Strecke zu fliegen, wodurch mit der Zeit Abweichungen der Position entstanden sind.

### 5.2.2 Neigungskompensation

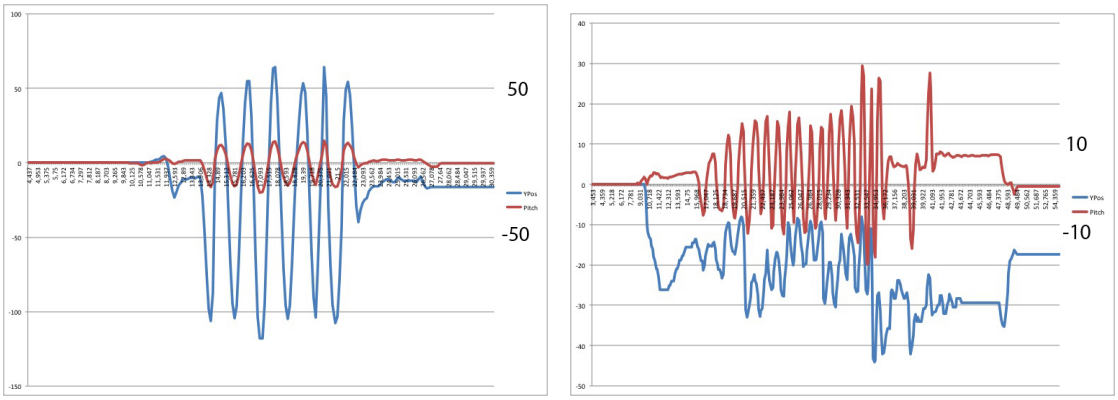
Auch bei der Neigungskorrektur wurde, wie bei der Höhenkompensation, eine Reihe von Messreihen aufgenommen. Es zeigt sich hier eine deutliche Verbesserung der Schwankungen, gleichzeitig aber auch eine Verschlechterung der gemessenen Endposition. Aus diesem Grund wurde die Neigungskorrektur beim Flug nicht verwendet, sondern nur die Höhenkompensation.




(a) Winkelkompensation deaktiviert (b) Winkelkompensation aktiviert

**Abbildung 5.8:** Vergleich X-Roll-Kompensation, Blau: Position, Rot: Winkel

Die Messungen wurden durchgeführt, indem der Quadrocopter auf einer Höhe von circa einem Meter mehrmals per Hand in die entsprechende Richtung und wie-




(a) Winkelkompensation deaktiviert (b) Winkelkompensation aktiviert

**Abbildung 5.9:** Vergleich Y-Pitch-Kompensation, Blau: Position, Rot: Winkel

der zurück gekippt wurde. Diese Prozedur wurde vier mal mit den entsprechenden Einstellungen wiederholt, um alle Möglichkeiten abzudecken.

Wie man in Abbildung 5.8 und Abbildung 5.9 gut erkennen kann, sind die Schwankungen der gedachten Position, bei gleicher Winkeländerung, im direkten Vergleich um rund den Faktor 10-15 geringer. Durch die Filterung der kleineren Winkeländerungen (siehe 4.3.2) wird jedoch ein Fehler erzeugt, der eine falsche Position erzeugt. Da während des Fluges selten sehr große Pitch- und Rollwinkel durch Richtungsänderungen entstehen (meistens durch Sensorfehler), wurde die Winkelkorrektur zumeist deaktiviert.

Es besteht jedoch mit dieser Implementierung eine Grundlage mit deren Hilfe eine genauere Korrektur ermöglicht werden kann.

Außerdem ist auch der Einfluss einer Winkeländerung des Yaw-Winkels kurz getestet worden, um bei Bedarf auch hier eine Kompensation einzufügen. Da bei anfänglichen Messungen allerdings kein klares Muster erkennbar war, wurde diese Korrektur nicht weiter untersucht und es wurden keine ausführlichen Tests unternommen.

# 6 Diskussion und Ausblick

Folgend werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick über eventuelle Optimierungen und damit verbundene Auswirkungen gegeben.

## 6.1 Diskussion

Zusammenfassend ist zu sagen, dass durch die zahlreichen Messungen und Versuche ein ansprechendes Ergebnis erzielt wurde. Besonders im Bereich der Landung und Positionsbestimmung wurden zum Teil große Verbesserungen festgestellt.

Die autonome Positionierung und Landung hat unter idealen Bedingungen zu einer deutlichen Genauigkeitssteigerung geführt. Die Landung erreicht eine Präzession von zum Teil unter sechs Zentimetern Abweichung und ist kann somit als punktgenau bezeichnet werden. Ungenauigkeiten und Fehler in der Lagesensorik stellen allerdings ein großes Problem dar und ermöglichen im schlimmsten Fall nur eine manuelle Landung, da das System seinen Zielpunkt nicht mehr autonom anfliegen kann. Auch die nicht näher getestete Landung in Intervallen, birgt noch die Möglichkeit einer Verbesserung in sich.

Die Veränderungen in der Positionsbestimmung mittels des Optischen-Fluss-Sensors haben auch - zumindest im Falle der Höhenkompensation - zu einer deutlichen Steigerung der Genauigkeit geführt. Somit ist es jetzt möglich, mit dem System auf verschiedenen Höhen zu operieren und trotzdem eine verlässliche Posi-

tionsangabe zu erhalten. Die Neigungskorrektur hingegen bring Vor- und Nachteile mit sich, die es abzuwägen gilt. Der Vorzug ist es, dass nun Schwankungen in der Position bei starken Winkeländerungen um den Faktor 10-15 schwächer ausfallen und somit eine stabilere Positionserfassung begünstigen. Ein großer Nachteil hingegen ist die, durch die Filterung von kleinen Winkeln (siehe 4.3.2), gesteigerte Fehlerrate in der Positionsbestimmung.

## 6.2 Ausblick

Insbesondere in Hinsicht auf die Implementierung einiger Methoden und Nutzerfreundlichkeit sind Verbesserungen möglich, auch eine Weiterführung dieser Arbeit ist denkbar. Folgend werden einige mögliche Verbesserungen aufgeführt.

Beispielsweise die leichtere und unkompliziertere Initialisierung und ein dadurch schnellerer Start können hinzugefügt werden. Dazu wäre es notwendig die Klassen der Objekterkennung und der Landung enger im Bereich des Laden und Scannen eines Bildes miteinander zu verknüpfen. Somit könnte man erreichen mit einigen wenigen Klicks das Programm zu starten und ein Landeziel zu suchen.

Des weiteren ist die Landung in Intervallen zu testen, analysieren und optimieren, um ein bestmögliches Ergebnis bei der Landung zu erzielen. Auch ist es hierbei wichtig, die Methoden zur manuellen und autonomen Landung voneinander zu trennen, um im Notfall eine Steuerung per Hand nicht zu behindern. Die aktuelle Implementierung der Arbeit greift, theoretisch, zu sehr in die bestehende Landesquenz ein und muss vor den ersten Tests getrennt werden.

Zuletzt ist zu nennen, dass eine weiterführende Arbeit das System weiter optimieren könnte, indem man dort versucht, den Ball durch am Boden erkannte Muster zu ersetzen. Somit könnte der Quadrocopter unabhängig von jeglichen Objekten an einem geeigneten Ort landen.

# 7 Literaturverzeichnis

[Lange et al. 2008] Lange, Sven ; Sünderhauf, Niko ; Protzel, Peter: Autonomous Landing for a Multirotor UAV Using Vision. In: *Chemnitz University of Technology* (2008)

[Reul 2013] Reul, Christian: *Implementierung und Evaluierung einer Objekterkennung für einen Quadrocopter*, University of Würzburg, Bachelorarbeit, 2013

[Rothe 2012] Rothe, Julian: *Implementierung und Evaluierung einer Höhenregelung für einen Quadrocopter*, University of Würzburg, Bachelorarbeit, 2012

[Wenzel et al. 2013] Wenzel, Karl E. ; Masselli, Andreas ; Scherer, Sebastian: Autonomous Flying Robots. In: *University of Tübingen* (2013)